Optimization for Machine Learning

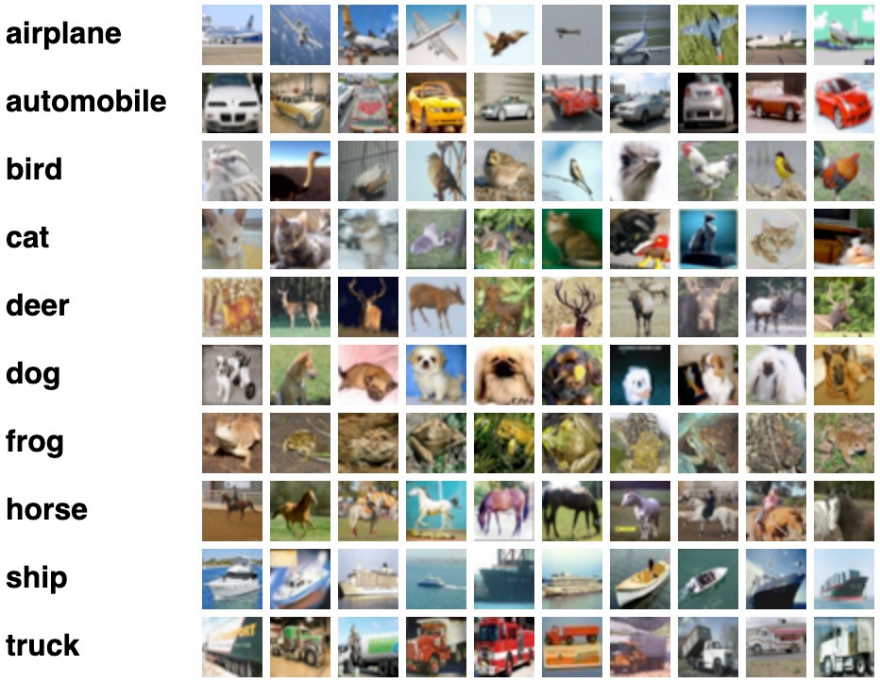
Instructor: Ashok Cutkosky

Logistics

- Course Description here: https://docs.google.com/document/d/1BGq8Txzhl9XIfAGHyzoBRKlFsah-qQJ7X_mijJbJju0
- Course notes available here: https://optmlclass.github.io/notes/optforml_notes.pdf
- Course notes will likely update a few times over the semester. First homework is available on blackboard. It is due on next Wednesday.
  - Math background exercises to practice writing proofs
  - I will grade this homework, but it will NOT count for your course grade. It is optional for you to get some idea of how to write proofs.
  - Submit on gradescope https://www.gradescope.com/courses/607283

Machine Learning Basics

Data/Observations → ML → Predictions/Conclusions



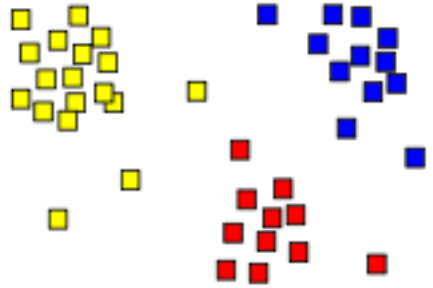
“Supervised learning”

Dog

Machine Learning Basics

Data/Observations → ML → Predictions/Conclusions

“unsupervised learning”



Two Steps for Machine Learning

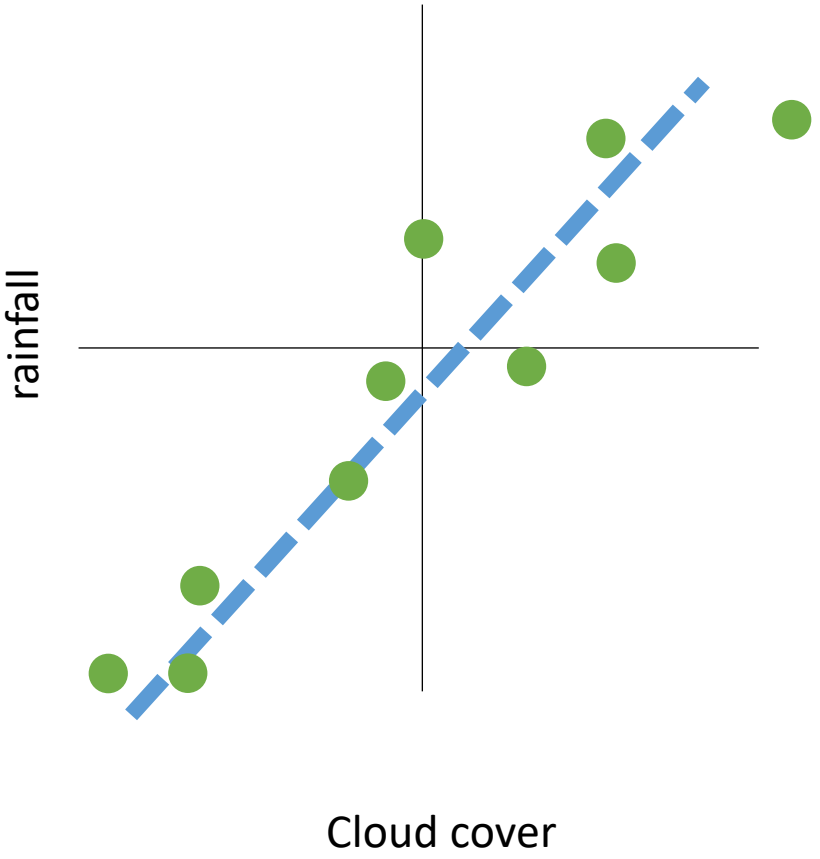
1. Choose a model.
   - In supervised learning, a model is a set of ways to generate predictions.
   - In unsupervised learning, a model might be a set of probability distributions to model the data, or a set of possible clusterings.
- Example: Linear regression.
  - Data: (x, y) $x \in \mathbb{R}^d$ "feature vector", $y \in \mathbb{R}$
  - Assume $y = x \cdot w_\star$ for some $w_\star \in \mathbb{R}^d$.
  - $w_\star$ is called the model **parameter**.
- In deep learning, choosing a model is sometimes called choosing an "architecture".

Two Steps for Machine Learning

2. Fit the model:
   - Gather a dataset of examples $z_1, \ldots, z_N$. (Training set).
   - Use training set to choose some $\hat{w}$ that has "good performance".
   - In linear regression: find $\hat{w}$ such that $y \approx x \cdot \hat{w}$.
- This is called "training"

Example: Linear Regression

- $z = (x, y)$
- (Step 1) Model $\hat{y} = xw^{(1)} + w^{(2)}$
  - Parameter $w = \left(w^{(1)}, w^{(2)}\right) \in \mathbb{R}^2$
- (Step 2) How should we choose w?

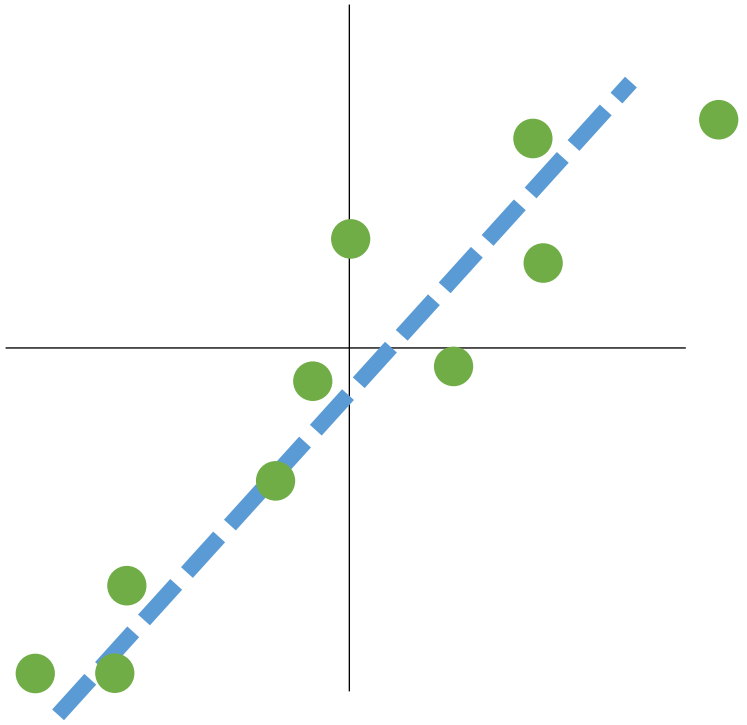


Two Steps for Machine Learning

- Most machine learning courses focus on step 1.
  - Can use domain knowledge.
  - A bit of a black art.
- This course focuses on understanding step 2.
  - Much more mathematical analysis possible today.
  - Still a lot of unanswered questions.
- Other ML courses at BU:
  - Statistical learning theory (possibly more mathy, lots of probability)
  - Reinforcement Learning
  - Deep learning (less mathy, more focus on step 1)

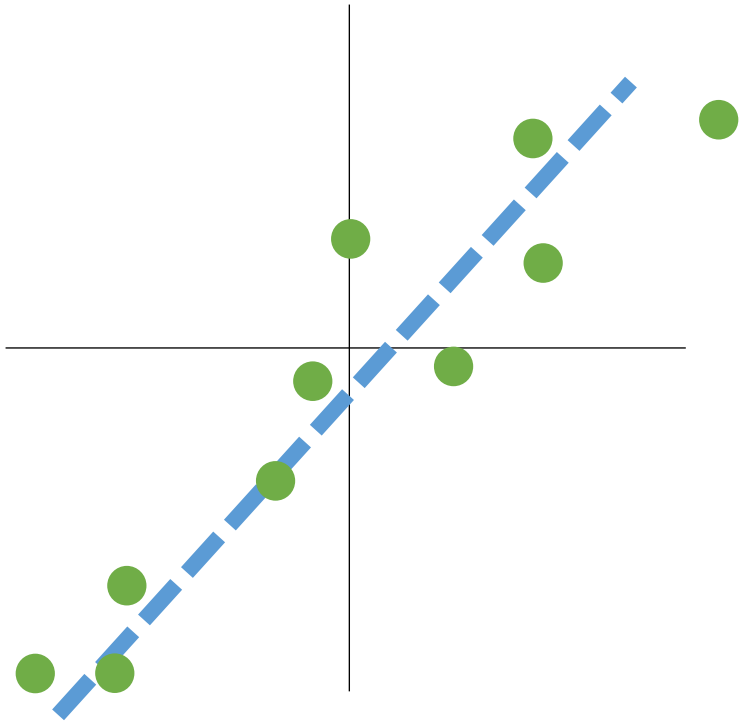
Example: Linear Regression

- $z = (x, y)$
- (Step 1) Model $\hat{y} = xw^{(1)} + w^{(2)}$
  - Parameter $w = \left(w^{(1)}, w^{(2)}\right) \in \mathbb{R}^2$
- (Step 2) How should we choose w?
- Classical statistics approach:
  - Assume $y = xw^{(1)} + w^{(2)} + r$
  - $r \sim N(0,1)$ (or some other kind of noise)
  - Find an unbiased estimate, or confidence bound, or maximum likelihood estimate…



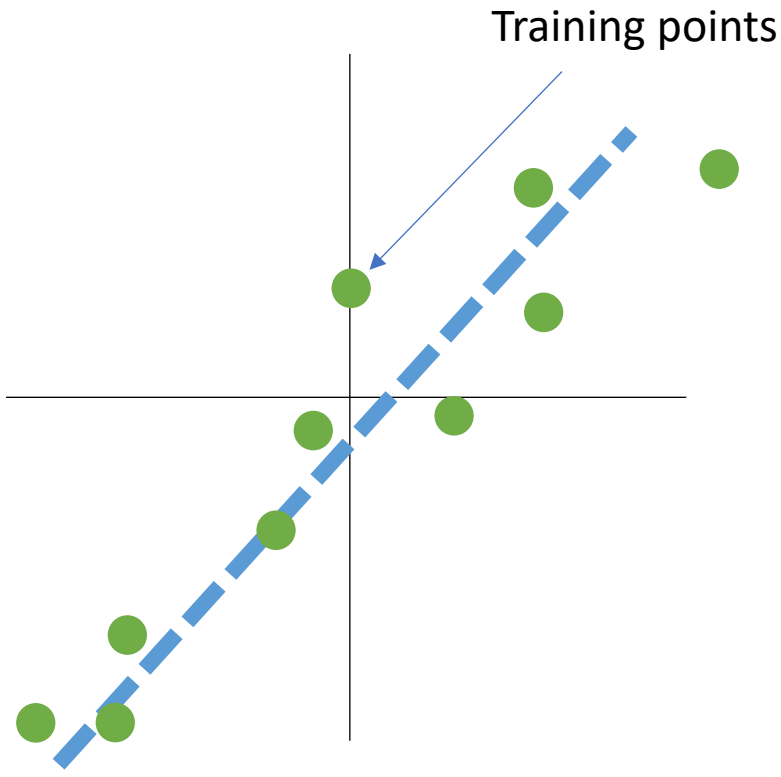
Example: Linear Regression

- $z = (x, y)$
- (Step 1) Model $\hat{y} = xw^{(1)} + w^{(2)}$
  - Parameter $w = \left(w^{(1)}, w^{(2)}\right) \in \mathbb{R}^2$
- (Step 2) How should we choose w?
- Machine Learning approach:
  - No assumption about y w.r.t. x, w
  - What is the *cost* of a prediction error?
  - Choose w to minimize this cost.



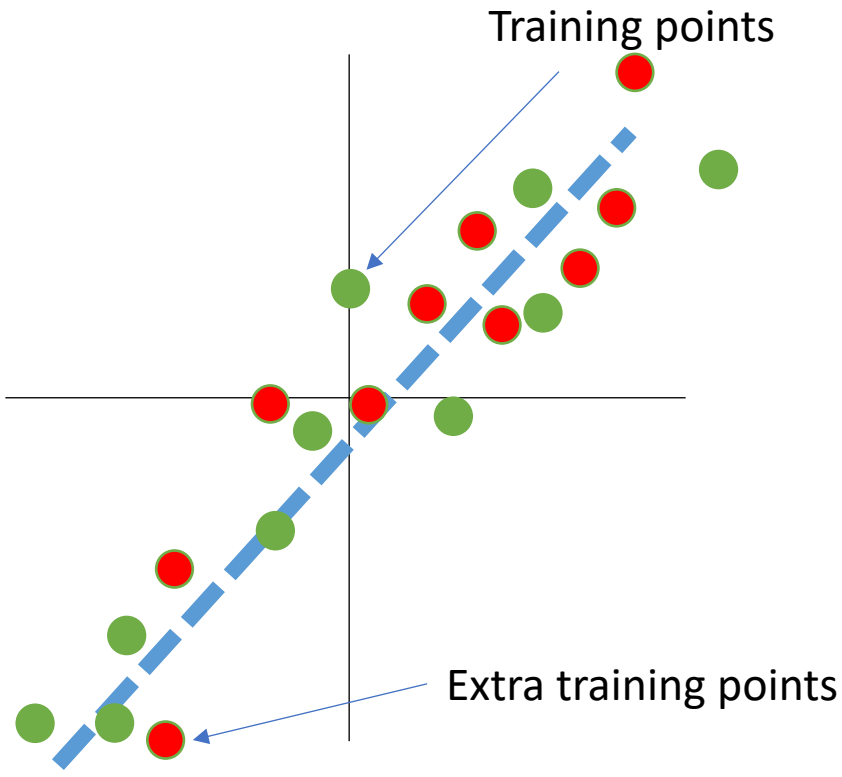
Training Sets

- To determine the "correct" value for w, we gather a *training set*.
- Training set = dataset of examples (i.e. x, y pairs) that we can use to measure the "quality" of a potential value for w.



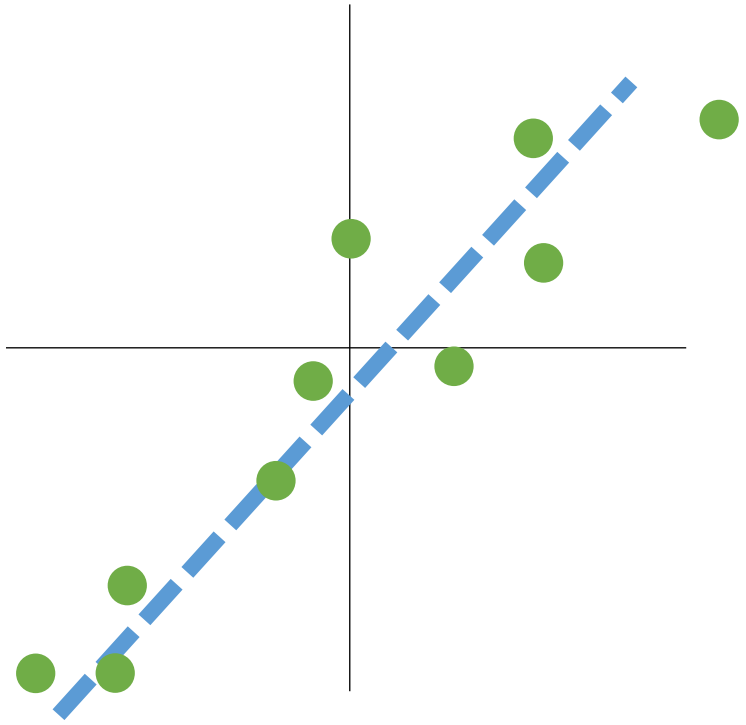
Training Sets

- To determine the "correct" value for w, we gather a *training set*.
- Training set = dataset of examples (i.e. x, y pairs) that we can use to measure the "quality" of a potential value for w.
- As we gather more training data, we should be able to find a better w value.



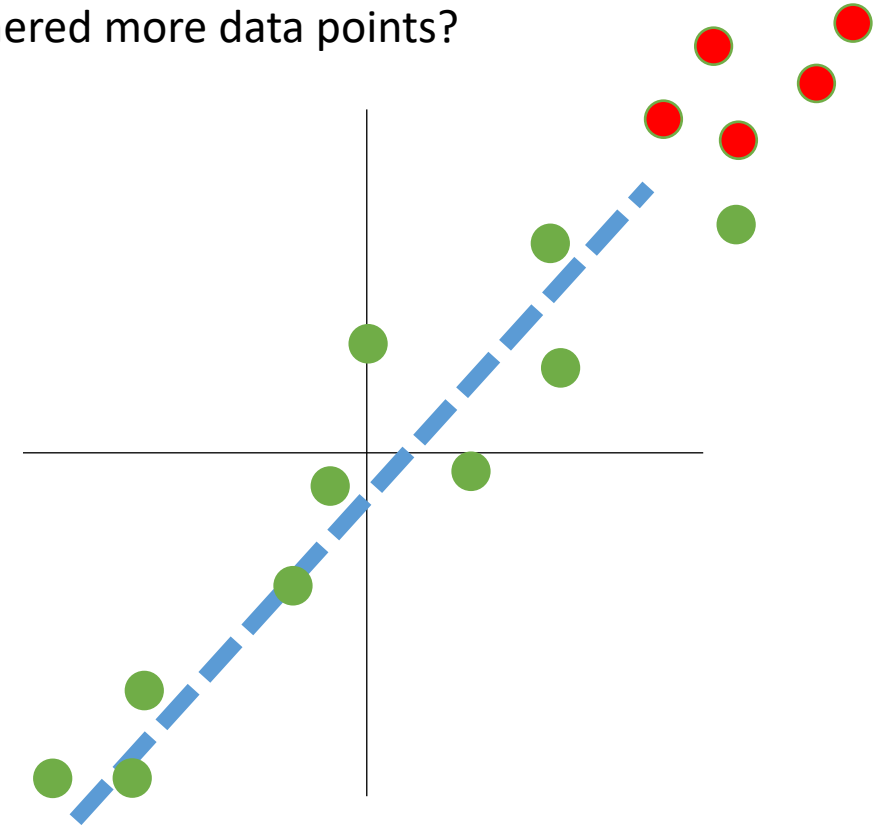
Independent and Identically Distributed Data

What would happen if we gathered more data points?



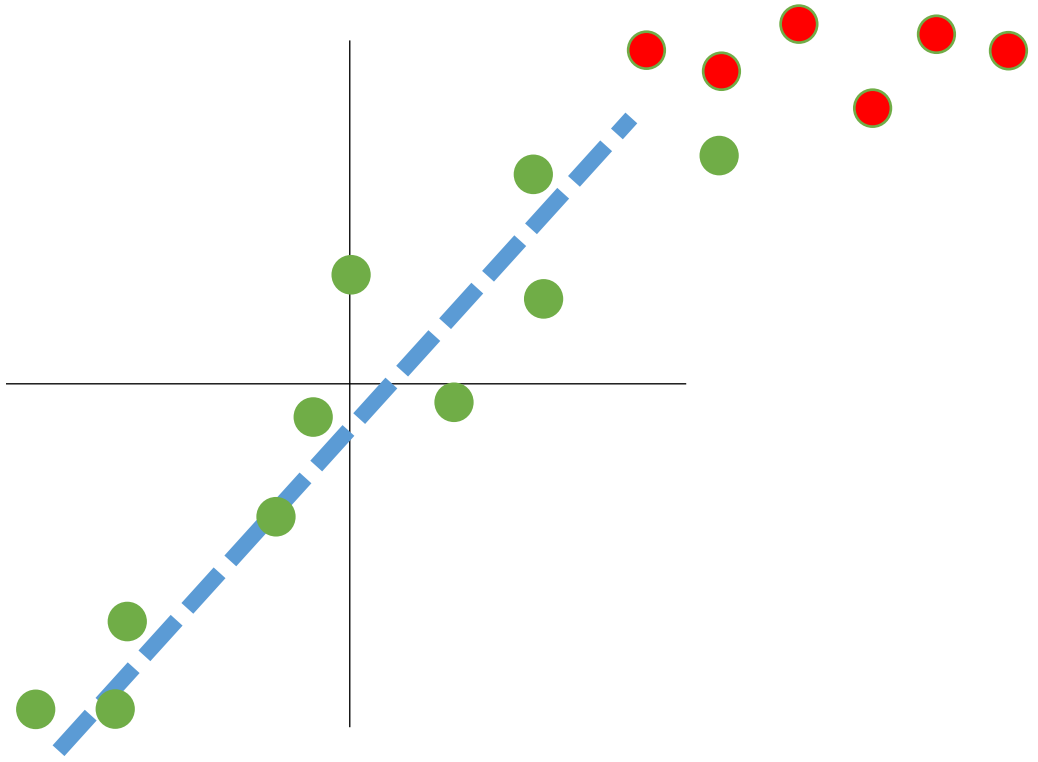
Independent and Identically Distributed Data

What would happen if we gathered more data points?



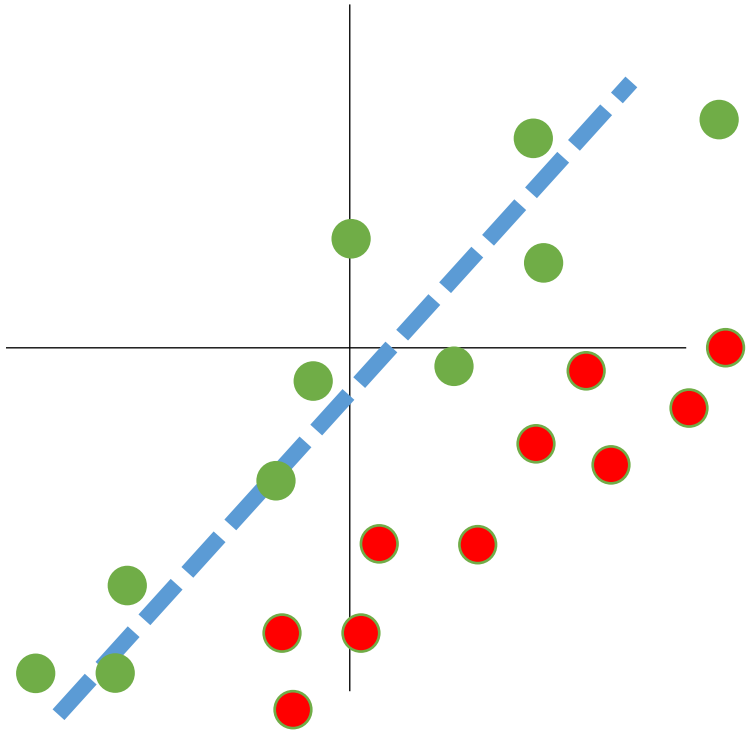
Independent and Identically Distributed Data

What would happen if we gathered more data points?



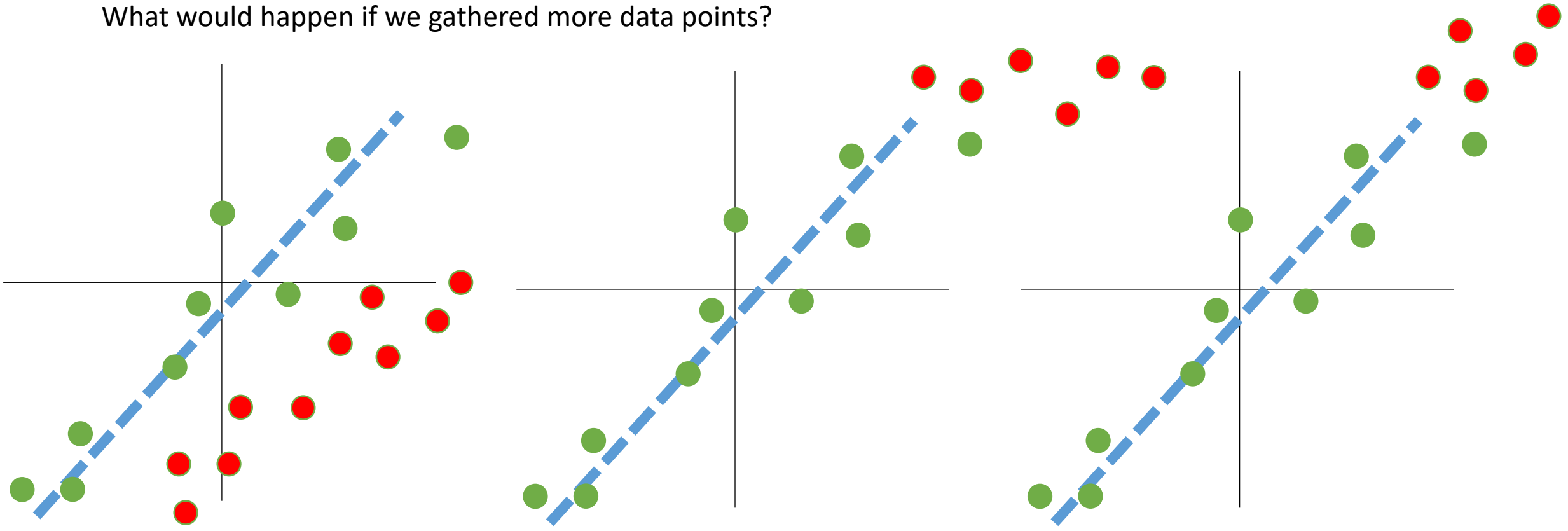
Independent and Identically Distributed Data

What would happen if we gathered more data points?



Independent and Identically Distributed Data

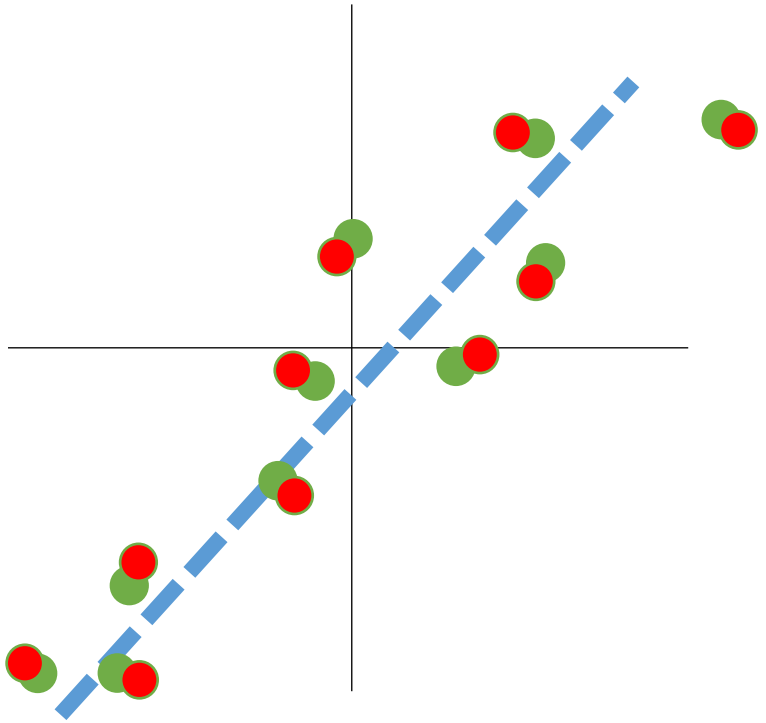
What would happen if we gathered more data points?



All of these examples are not *identically distributed*

Independent and Identically Distributed Data

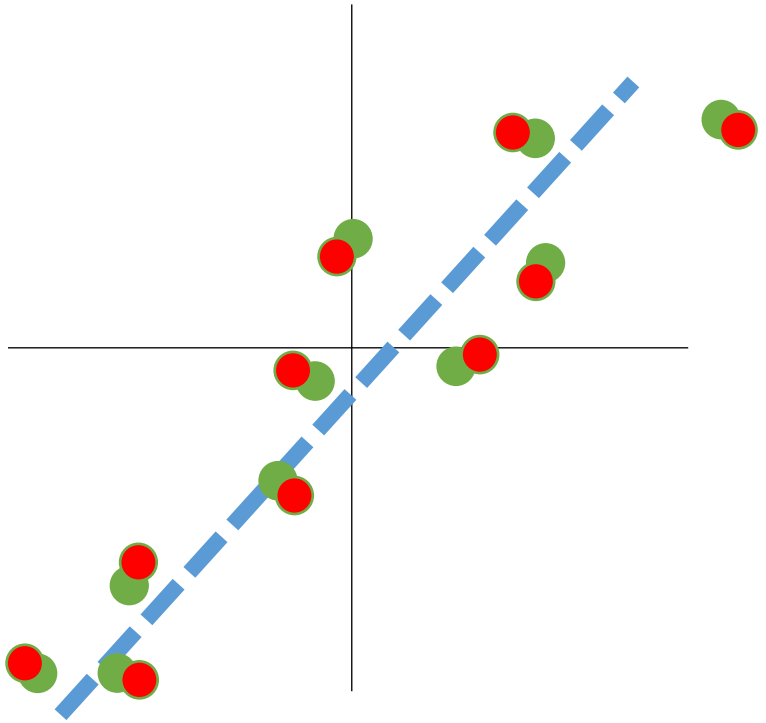
What would happen if we gathered more data points?



Adding the red points to the training set is not helpful!

Independent and Identically Distributed Data

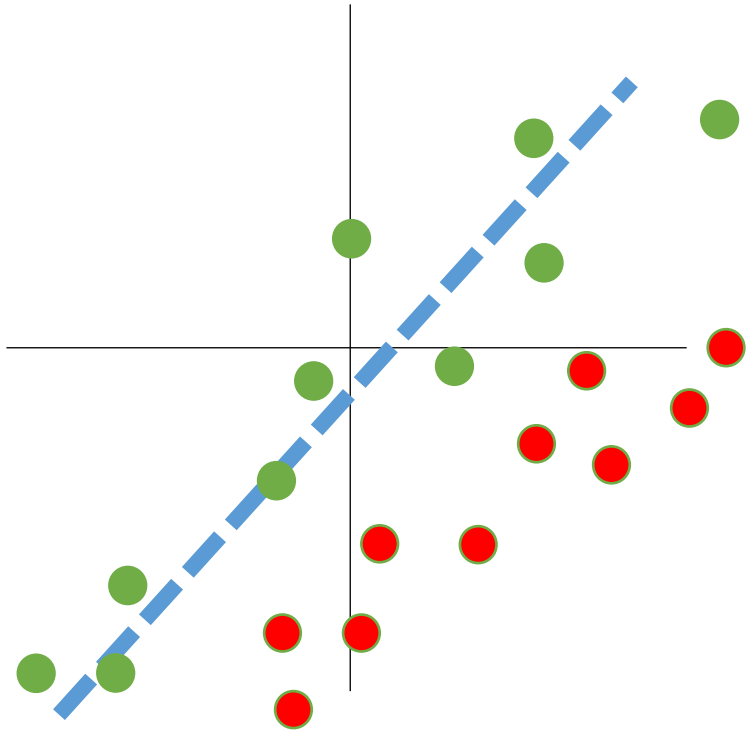
What would happen if we gathered more data points?



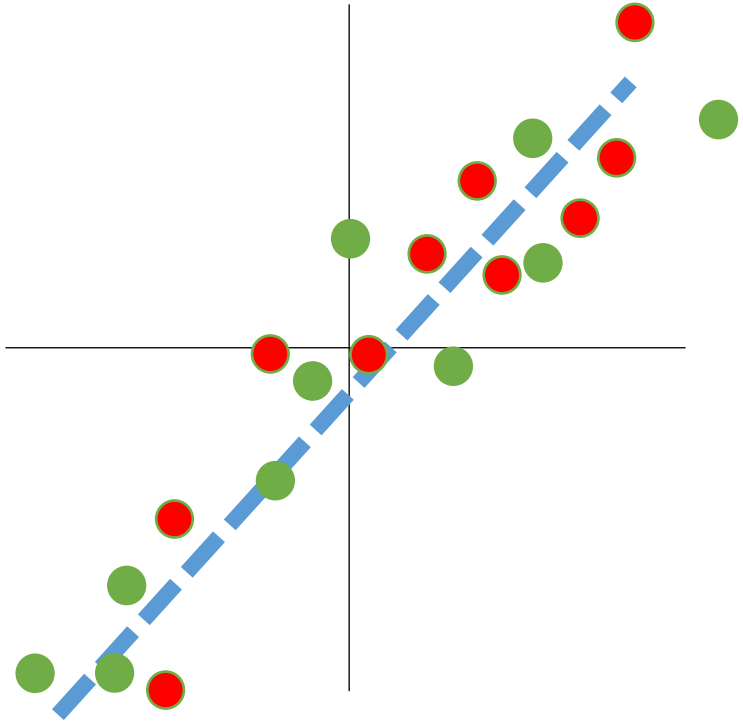
This example is not *independent*

Independent and Identically Distributed Data

What would happen if we gathered more data points?



Identically distributed: Current data describes future data



Independent: new data points provide new information

Formal Description of Machine Learning

- A distribution over "examples" $z \sim P_z$.
- A training set $z_1, \ldots, z_N$.
  - Critical assumption: each z_i is sampled i.i.d. from P_z. (independently and identically distributed).
- A **loss function** $\ell(w, z)$. This defines a "true" or "population" loss:

$$\mathcal{L}(w) = E_z[\ell(w, z)]$$

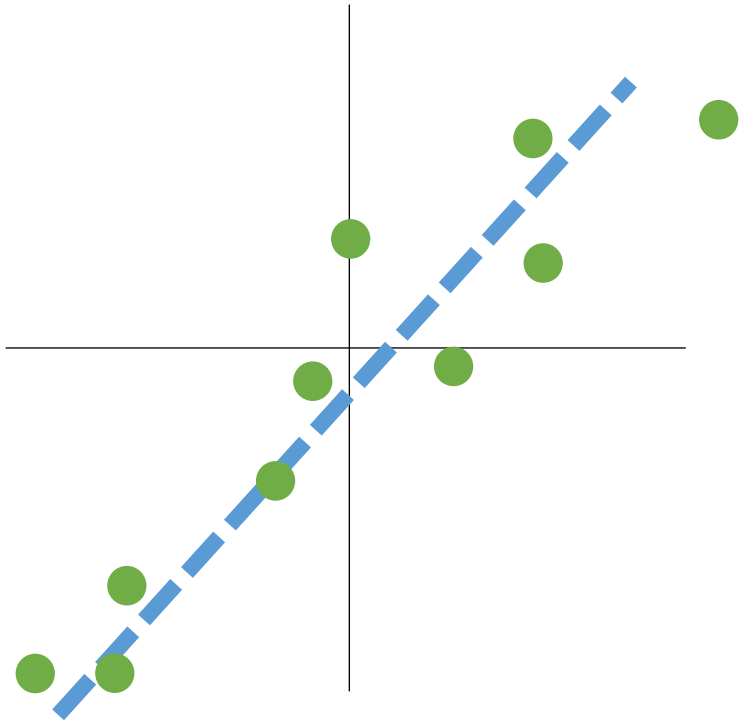
- The task is to find $\widehat{w}$ that minimizes $\mathcal{L}(w)$.
  - This is an optimization problem!

Linear Regression in the Formal Setting

- $z = (x, y)$
- Model $\hat{y} = xw^{(1)} + w^{(2)}$
  - Parameter $w = \left(w^{(1)}, w^{(2)}\right) \in \mathbb{R}^2$
- Can choose loss function.
- Square loss:

$$\ell(w, z) = (y - \hat{y})^2$$
$$= \left(y - xw^{(1)} - w^{(2)}\right)^2$$

- Absolute loss: $\ell(w, z) = |y - \hat{y}|$

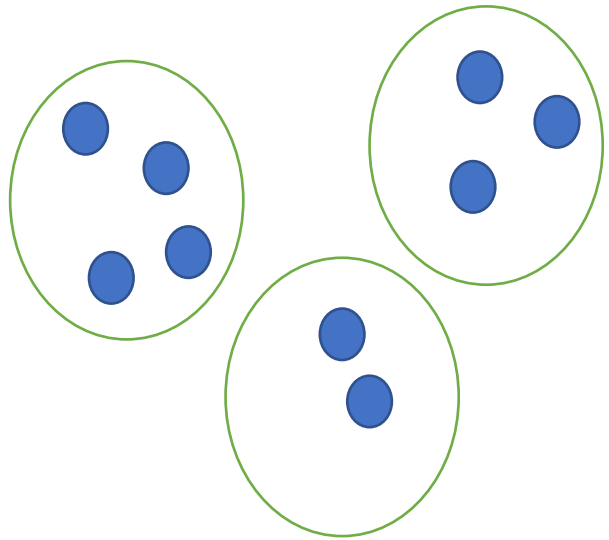


Clustering in the Formal Setting

- $z \in \mathbb{R}^2$
- $w = [c_1,\ c_2, c_3]$
  - c_i is the center of the ith cluster.
- Loss function:

$$\ell(w, z) = \min_i ||z - c_i||$$

- Other possibilities also reasonable.



Formal Description Cont.

- Choice of loss function ℓ abstracts out the "step 1" of machine learning.
- Supervised learning setting: $z = (x,\ y)$, trying to predict y from input x.
- Set $\hat{y} = f(w, x)$.
  - f could be anything, like a linear model, a decision tree, or a neural network.
- "prediction error function" $error(y, \hat{y})$

$$\ell(w, z) = error(y, f(w, x))$$

- Often the error function is also called a loss function.

Stochastic Optimization

$$\mathcal{L}(w) = E_z[\ell(w, z)]$$

- Want low suboptimality: $\mathcal{L}(\widehat{w}) - \mathcal{L}(w^\star), \quad w^\star = argmin\, \mathcal{L}(w)$
- The distribution for z is *unknown*.
- Our only information about the distribution is a training set $z_1, \ldots, z_N$ sampled i.i.d. from this distribution.
- How can we use this information to minimize $\mathcal{L}$?

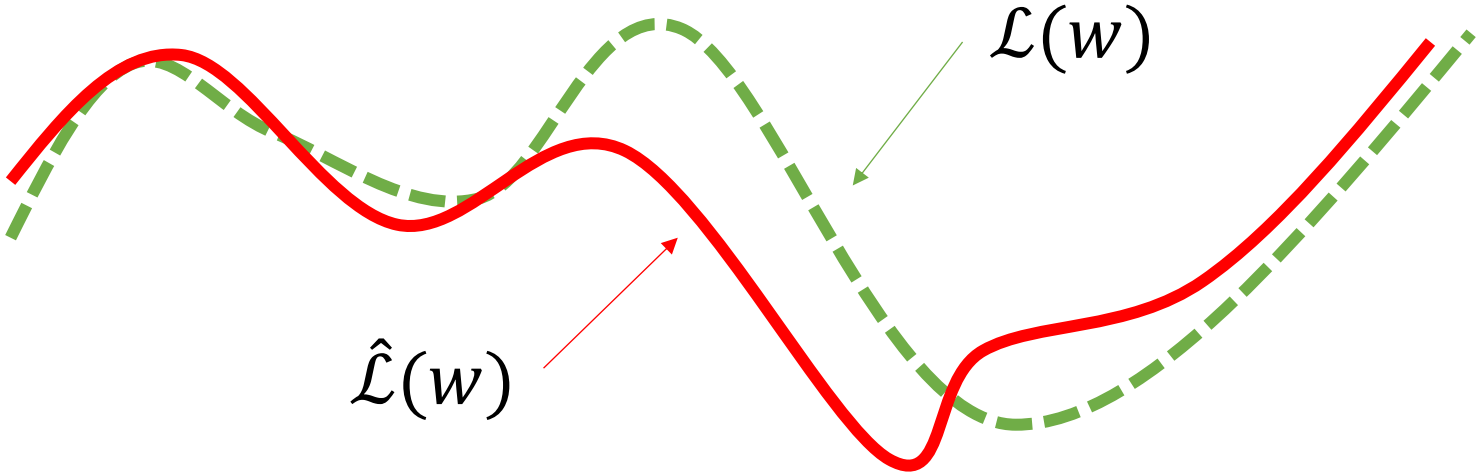
Classical approach: Empirical Risk Minimization (ERM)

- Approximate $\mathcal{L}$ as:

"empirical risk"

$$\mathcal{L}(w) \approx \hat{\mathcal{L}}(w) = \frac{1}{N}\sum_{i=1}^{N} \ell(w, z_i)$$

- Intuition: averages converge to the expectation.



ERM

- The empirical risk function is completely known in principle – NOT stochastic.
- Optimize it with some algorithm (still a hard problem!).
  - computational expensive to evaluate the empirical risk (why?)
- Get $\widehat{w}_{ERM} \approx argmin\, \hat{\mathcal{L}}(w)$
- Hope that $\widehat{w}_{ERM} \approx argmin\, \hat{\mathcal{L}}(w)$

ERM decomposition

- Basic formula for ERM:

$$\mathcal{L}(\widehat{w}_{ERM}) - \mathcal{L}(w^{\star})$$

ERM decomposition

- Basic formula for ERM:

$$\mathcal{L}(\widehat{w}_{ERM}) - \mathcal{L}(w^\star)$$
$$= \mathcal{L}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(\widehat{w}_{ERM}) + \hat{\mathcal{L}}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(w^\star_{ERM}) + \hat{\mathcal{L}}(w^\star_{ERM})$$
$$- \hat{\mathcal{L}}(w^\star) + \hat{\mathcal{L}}(w^\star) - \mathcal{L}(w^\star)$$

ERM decomposition

- Basic formula for ERM:

$$\begin{aligned}
&\mathcal{L}(\widehat{w}_{ERM}) - \mathcal{L}(w^\star) \\
&= \mathcal{L}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(\widehat{w}_{ERM}) + \hat{\mathcal{L}}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(w^\star_{ERM}) + \hat{\mathcal{L}}(w^\star_{ERM}) \\
&- \hat{\mathcal{L}}(w^\star) + \hat{\mathcal{L}}(w^\star) - \mathcal{L}(w^\star) \\
&= \mathcal{L}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(\widehat{w}_{ERM}) + \hat{\mathcal{L}}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(w^\star_{ERM}) \\
&\quad + \hat{\mathcal{L}}(w^\star_{ERM}) - \hat{\mathcal{L}}(w^\star) + \hat{\mathcal{L}}(w^\star) - \mathcal{L}(w^\star)
\end{aligned}$$

- The term $\hat{\mathcal{L}}(\widehat{w}_{ERM}) - \hat{\mathcal{L}}(w^\star_{ERM})$ is controlled by the optimization algorithm.
- The other terms cause *overfitting*.

ERM is a special stochastic optimization problem

- ERM problem is also a *finite-sum objective*:

$$\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N} \ell(w, z_i)$$

- Is also stochastic optimization with special distribution $z \sim \hat{P}_z$:

$$\hat{\mathcal{L}}(w) = E_{z\sim\hat{P}_z}[\ell(w, z)]$$

- What is $\hat{P}_z$?
  - Uniform distribution over $z_1, \ldots, z_N$.
- Even though we *can* compute the exact ERM loss $\hat{\mathcal{L}}$, it will turn out to be faster to pretend that we cannot and treat ERM as stochastic optimization.

Convex and Non-Convex problems

- Many classical machine learning models produce losses that are *convex*.
  - SVMs, linear regression/classification.
- Huge amount of literature on how to optimize convex functions.
- Non-convex problems are much harder.
  - Neural networks are non-convex

First homework due next wednesday

- Exercises in probability and matrices.
- Intended to refresh skills.
- The appendix of the course notes has some background material that might be useful.
  - Markov inequality:
    - For any non-negative random variable X and value y, $P[X \geq y] \leq \frac{E[X]}{y}$
  - Jensen inequality:
    - For any convex function f and random variable X, $f(E[X]) \leq E[f(x)]$