

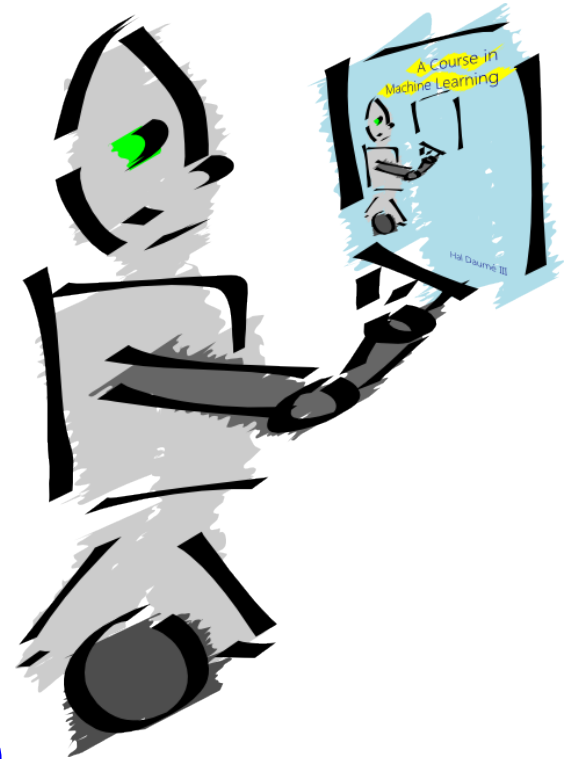
Machine Learning I

HAL DAUMÉ III, UMD

me@hal3.name

<http://hal3.name>

[@haldaume3](#)



Credit: many slides due to Marine Carpuat (UMD)

Loosely in parts based on A Course in Machine Learning (ciml.info)

What is this course about?

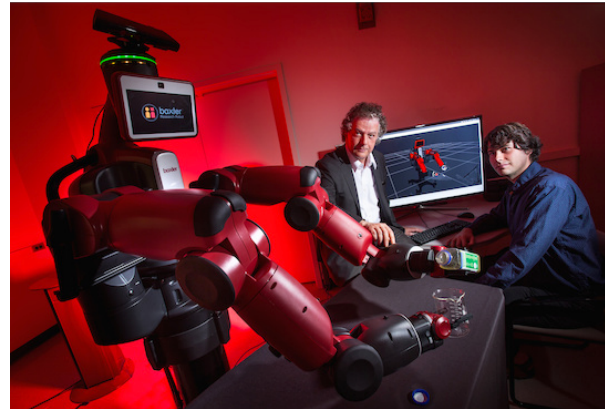
Machine learning studies **algorithms for learning to do stuff**

- By finding (and exploiting) patterns in data
- Sometimes in ways we'd rather they didn't
- Theory helps us understand this!

What can we do with machine learning?



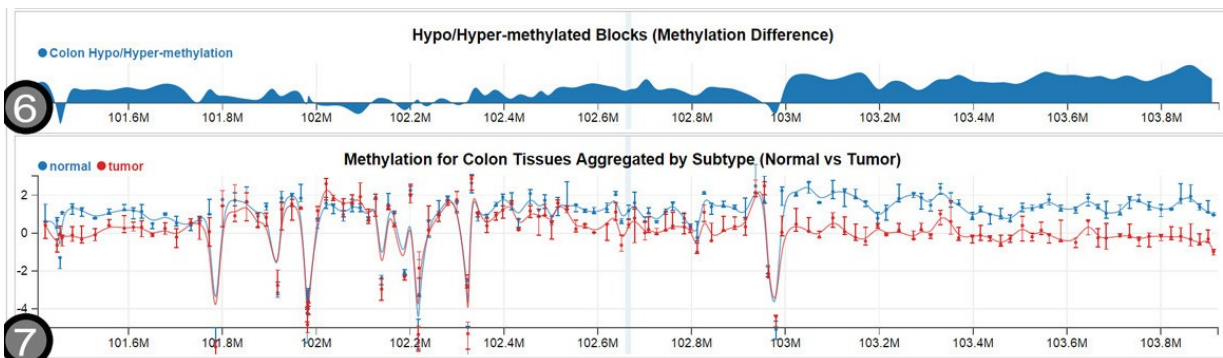
Analyze text & speech



Teach robots how to cook from youtube videos

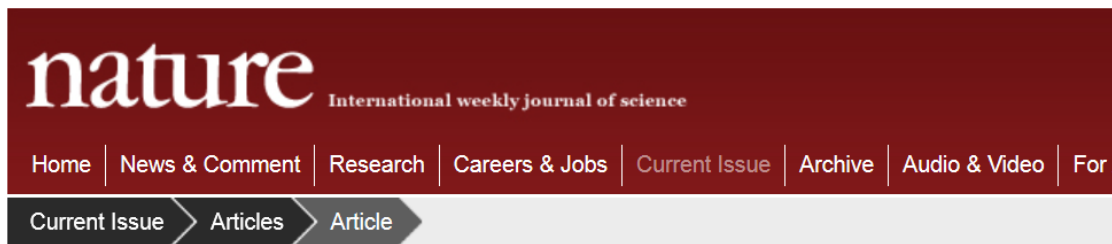


Recognize objects in images



Analyze genomics data

Sometimes machines even perform better than humans!



NATURE | ARTICLE

日本語要約

Mastering the game of Go with deep neural networks and tree search

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

[Affiliations](#) | [Contributions](#) | [Corresponding authors](#)

Nature **529**, 484–489 (28 January 2016) | doi:10.1038/nature16961

Received 11 November 2015 | Accepted 05 January 2016 | Published online 27 January 2016



Question Answering system beats Jeopardy champion Ken Jennings at Quiz bowl!

Machine Learning

- Paradigm: “Programming by example”
 - Replace “human writing code” with “human supplying data”
- Most central issue: generalization
 - How to abstract from “training” examples to “test” examples?

A growing and fast moving field

- Broad applicability
 - Finance, robotics, vision, machine translation, medicine, etc.
- Close connection between theory and practice
- Open field, lots of room for new work!

Course Goals

- By the end of the week, you should be able to
 - Frame simple problems as ML problems (if appropriate)
 - Understand when generalization is and isn't possible
 - Find many examples in the real world of systems that do not generalize well and explain why
 - Have access to tools for building your own ML solutions to problems
- This course is **not**
 - A survey of ML algorithms
 - A tutorial on ML toolkits such as Weka, TensorFlow, ...

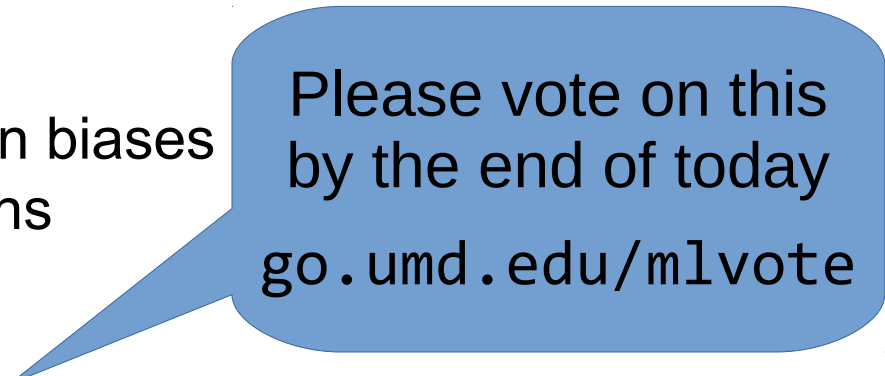
Topics

Today: Foundations of Supervised Learning

- Inductive bias and generalization
- Linear models and gradient descent
- Overfitting and the bias/variance trade-off

Next: Bias in Machine Learning

- Neural networks
- Machine learning replicates human biases
- Sample selection bias and solutions



Please vote on this
by the end of today
go.umd.edu/mlvote

Finally: Somewhat up in the air

- Attacking machine learning systems & defending them
- Bias in language reflected in learned systems
- Understanding the behavior of learned systems
- Building learning systems, beyond differentiability

Who am I and how did I get here?

- Grew up in LA, parents both did marketing research
- High school
 - math, Latin, programming
 - new goal in life: HS math teacher
- College
 - discrete math & creative writing, then Chris Quirk at LTI
 - new goal in life: teach undergrads math
- Grad school
 - CS, natural language processing ... wait, but *math*!
 - machine learning “revolution”
 - new goal in life: teach grad students
- - (1) how can we get computers to learn language through natural interaction with people/users?
 - (2) how can we do this in a way that promotes fairness, transparency and explainability in the learned models?

I'm a real person too....

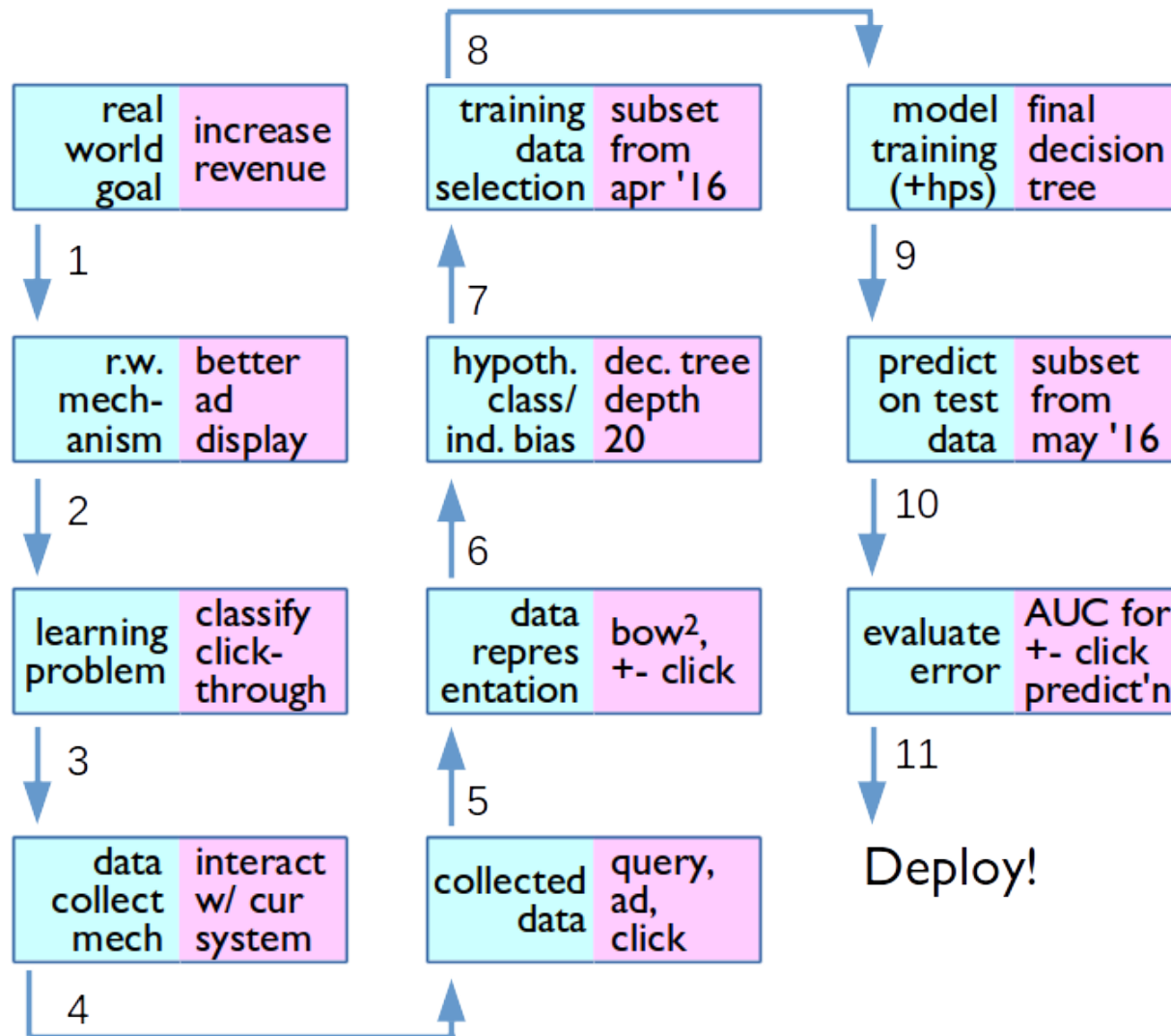


Today's topics

What does it mean to “learn by example”?

- Classification tasks
- Inductive bias
- Formalizing learning

Typical Design Process for an ML Application



Classification tasks

- How would you write a program to distinguish a picture of me from a picture of someone else?
- Provide examples pictures of me and pictures of other people and let a **classifier** learn to distinguish the two.

Classification tasks

- How would you write a program to distinguish a **sentence** is **grammatical** or **not**?
- Provide examples of **grammatical** and **ungrammatical sentences** and let a **classifier** learn to distinguish the two.

Classification tasks

- How would you write a program to distinguish **cancerous cells** from **normal cells**?
- Provide examples of **cancerous** and **normal cells** and let a **classifier** learn to distinguish the two.

Classification tasks

- How would you write a program to distinguish **cancerous cells** from **normal cells**?
- Provide examples of **cancerous** and **normal cells** and let a **classifier** learn to distinguish the two.

Let's try it out...

- Your task: learn a classifier to distinguish class A from class B from examples

- Examples of class A:



- Examples of class B



Let's try it out...

- ✓ learn a classifier from examples
- Now: predict class on new examples using what you've learned













Key ingredients needed for learning

- Training vs. test examples
 - Memorizing the training examples is not enough!
 - Need to generalize to make good predictions on test examples
- Inductive bias
 - Many classifier hypotheses are plausible
 - Need assumptions about the nature of the relation between examples and classes

What if I told you...

B



B



A



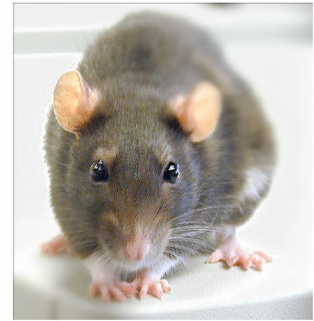
B



A



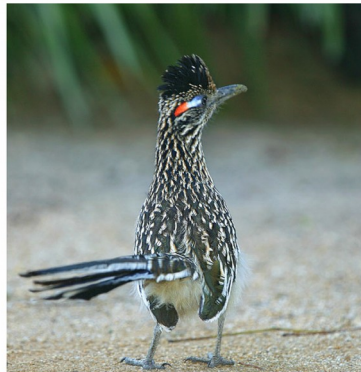
B



B



A



A



Machine Learning as Function Approximation

- Problem setting
 - Set of possible instances X
 - Unknown target function $f: X \rightarrow Y$
 - Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \dots (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Formalizing induction: Loss Function

-

$l(y, f(x))$ where y is the truth and $f(x)$ is the system's prediction

$$\text{e.g. } l(y, f(x)) = \begin{cases} 0 & \text{if } y = f(x) \\ 1 & \text{otherwise} \end{cases}$$

Captures our notion of what is important to learn

Formalizing induction: Data generating distribution

- - Where does the data come from?
 - Data generating distribution
 - A probability distribution D over (x, y) pairs
 - We don't know what D is!
 - We only get a random sample from it: our training data

Formalizing induction: Expected loss

- - f should make good predictions
 - as measured by loss l
 - on **future** examples that are also drawn from D
 - Formally
 - ε , the expected loss of f over D with respect to l should be small

$$\varepsilon \triangleq \mathbb{E}_{(x,y) \sim D} \{l(y, f(x))\} = \sum_{(x,y)} D(x, y) l(y, f(x))$$

Formalizing induction:

Training error

-
- We can't compute expected loss because we don't know what D is
- We only have a sample of D
 - training examples $\{(x^{(1)}, y^{(1)}), \dots (x^{(N)}, y^{(N)})\}$
- All we can compute is the training error

$$\hat{\varepsilon} \triangleq \sum_{n=1}^N \frac{1}{N} l(y^{(n)}, f(x^{(n)}))$$

Formalizing Induction

- Given
 - a loss function l
 - a sample from some **unknown** data distribution D
- Our task is to compute a function f that has low expected error over D with respect to l .

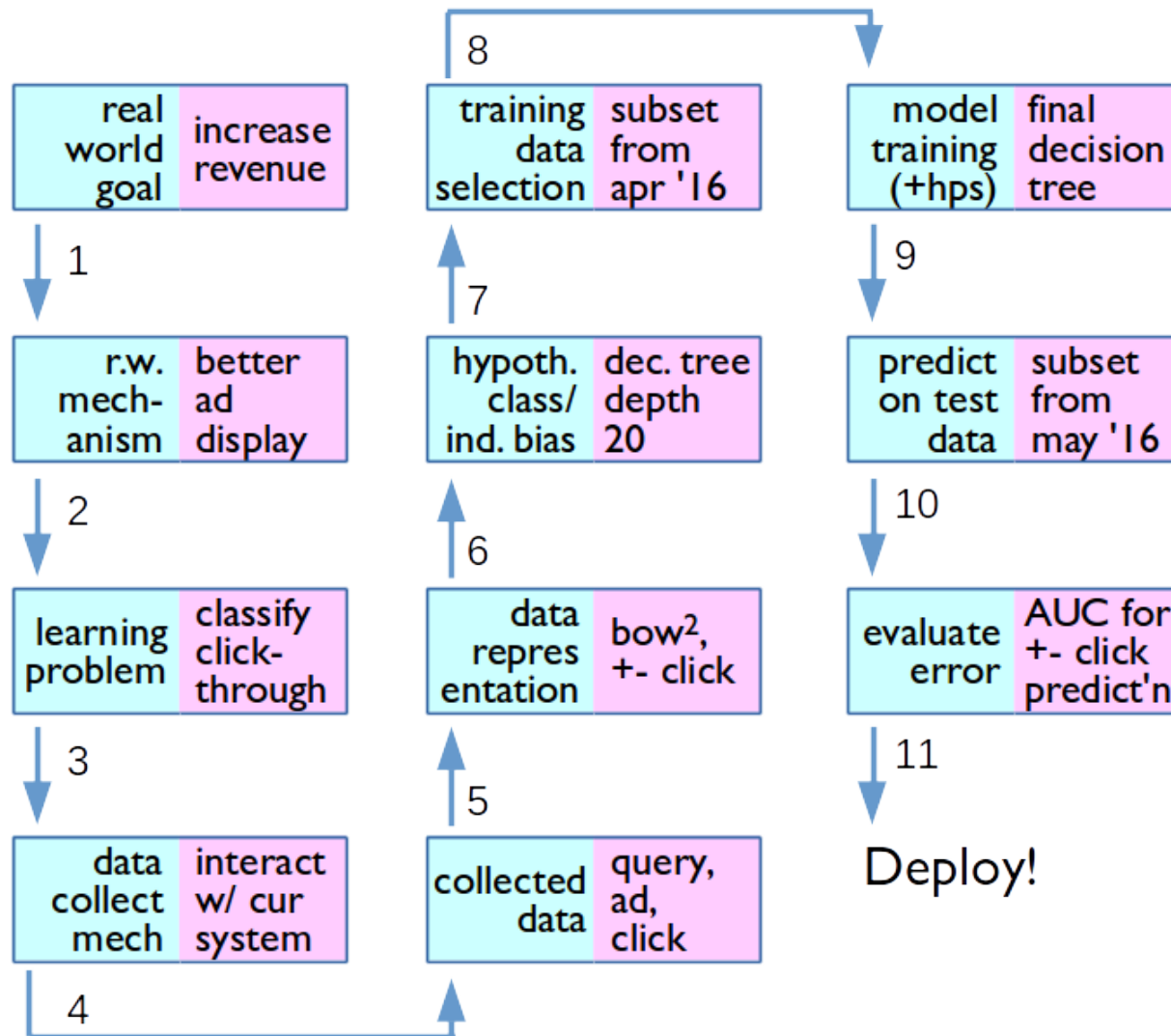
$$\mathbb{E}_{(x,y) \sim D} \{l(y, f(x))\} = \sum_{(x,y)} D(x, y) l(y, f(x))$$

Recap: introducing machine learning

What does “learning by example” mean?

- Classification tasks
- Learning requires examples + inductive bias
- Generalization vs. memorization
- Formalizing the learning problem
 - Function approximation
 - Learning as minimizing expected loss

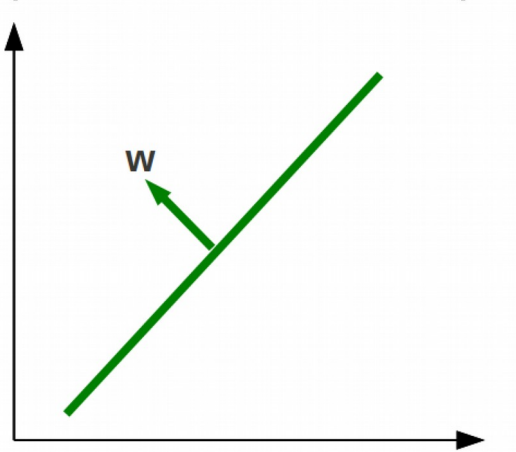
Typical Design Process for an ML Application



Topics

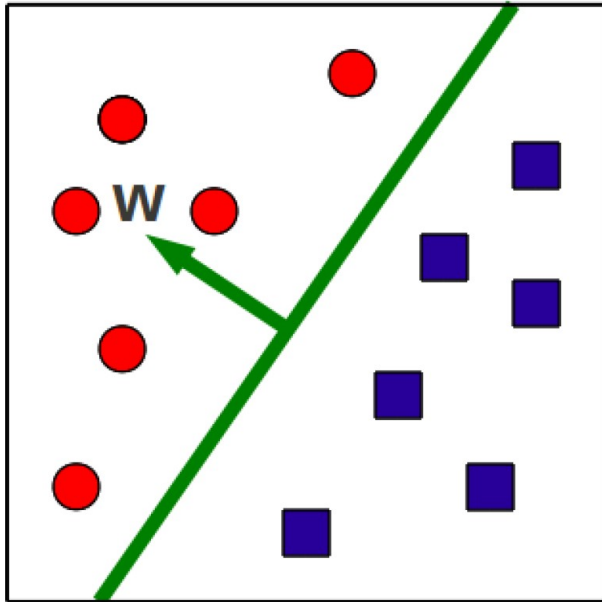
- Linear Models
 - Loss functions
 - Regularization
- Gradient Descent
- Calculus refresher
 - Convexity
 - Gradients

Geometry concept: Hyperplane



- Separates a D-dimensional space into two half-spaces
- Defined by an outward pointing normal vector
 - is **orthogonal** to any vector lying on the hyperplane
- Hyperplane passes through the origin, unless we also define a **bias** term b

Binary classification via hyperplanes



-
- A classifier is a hyperplane (w, b)
- At test time, we check on what side of the hyperplane examples fall
$$\hat{y} = \text{sign}(w^T x + b)$$
- This is a **linear classifier**
 - Because the prediction is a linear combination of feature values x

Class y

What real data looks like.

Example

1 robocop is an intelligent science fiction thriller and social satire , one with class and style . the film , set in old detroit in the year 1991 , stars peter weller as murphy a lieutenant on the city's police force . 1991's detroit is a police department in a city concepts are threatening a group of , a savage group of city . [...] do the city ? they have resulted it into a live action effects , embarrassing writing and kid-friendly slapstick . wasn't mr . magoo enough , people ? obviously not . inspector gadget is not what i would call ideal family entertainment . [...]

How would you define input vectors x to represent each example? What features would you use?

TASK: BINARY CLASSIFICATION

Given:

1. An input space \mathcal{X}
2. An unknown distribution \mathcal{D} over $\mathcal{X} \times \{-1, +1\}$

Compute: A function f minimizing: $\mathbb{E}_{(x,y) \sim \mathcal{D}} [f(\mathbf{x}) \neq y]$

Learning a Linear Classifier as an Optimization Problem

**Objective
function**

$$\min_{\mathbf{w}, b} L(\mathbf{w}, b)$$

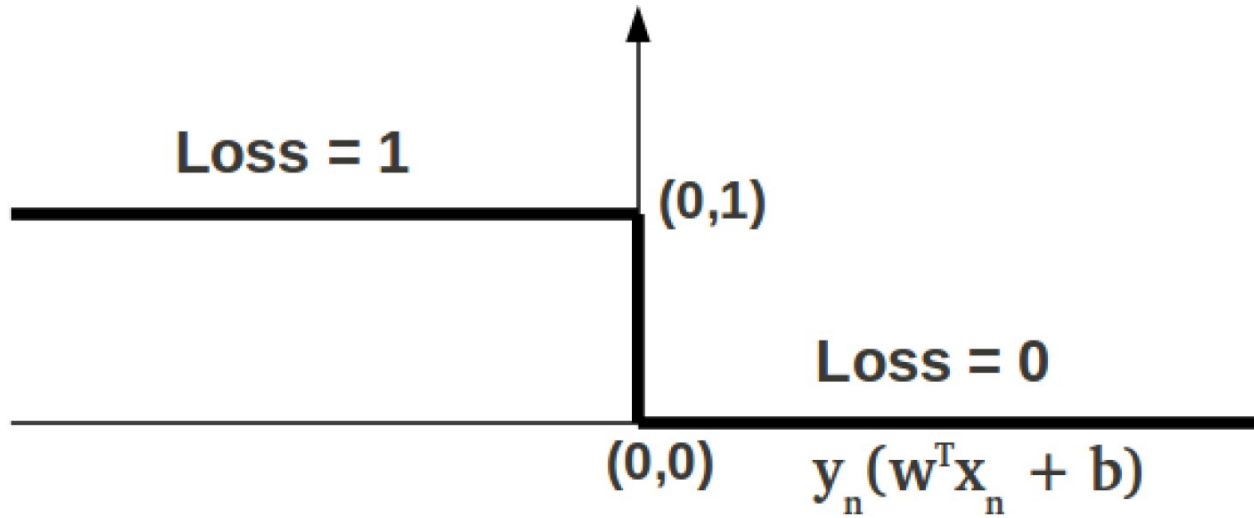
Loss function

measures how well
classifier fits training
data

Regularizer

prefers solutions
that generalize
well

The 0-1 Loss



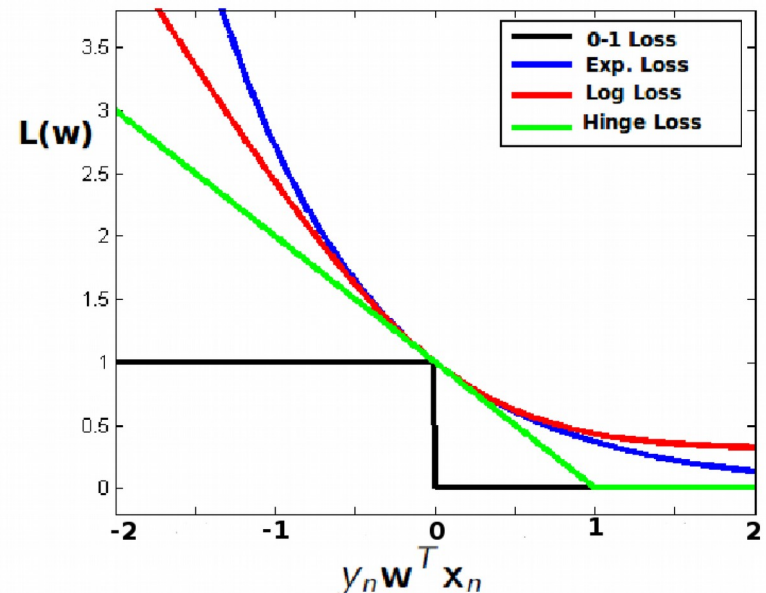
- Small changes in w, b can lead to big changes in the loss value
- 0-1 loss is non-smooth, non-convex

Calculus refresher:

Smooth functions, convex functions

Approximating the 0-1 loss with surrogate loss functions

- Examples (with $b = 0$)
 - Hinge loss $[1 - y_n \mathbf{w}^T \mathbf{x}_n]_+ = \max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\}$
 - Log loss $\log[1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)]$
 - Exponential loss $\exp(-y_n \mathbf{w}^T \mathbf{x}_n)$
- All are convex upper-bounds on the 0-1 loss



Casting Linear Classification as an Optimization Problem

Objective function

Loss function
measures how well classifier fits training data

Regularizer
prefers solutions that generalize well

$$\min_{\mathbf{w}, b} L(\mathbf{w}, b) = \min_{\mathbf{w}, b} \sum_{n=1}^N \mathbb{I}(y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

$\mathbb{I}(\cdot)$ Indicator function: 1 if (\cdot) is true, 0 otherwise
The loss function above is called the 0-1 loss

The regularizer term

- Goal: find simple solutions (inductive bias)
- Ideally, we want most entries of \mathbf{w} to be zero, so prediction depends only on a small number of features.
- Formally, we want to minimize:

$$R^{cnt}(\mathbf{w}, b) = \sum_{d=1}^D \mathbb{I}(w_d \neq 0)$$

- That's NP-hard, so we use approximations instead.
 - E.g., we encourage w_d 's to be small

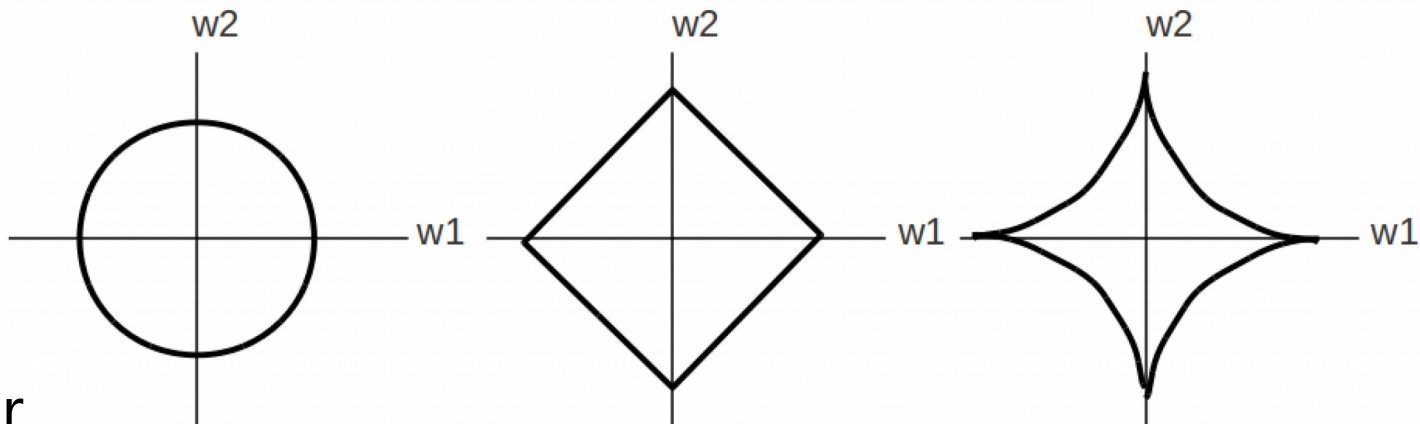
Norm-based Regularizers

- l_p norms can be used as regularizers

$$||\mathbf{w}||_2^2 = \sum_{d=1}^D w_d^2$$

$$||\mathbf{w}||_1 = \sum_{d=1}^D |w_d|$$

$$||\mathbf{w}||_p = \left(\sum_{d=1}^D w_d^p \right)^{1/p}$$



Contour
plots for

$p = 2$

$p = 1$

$p < 1$

Casting Linear Classification as an Optimization Problem

Objective function

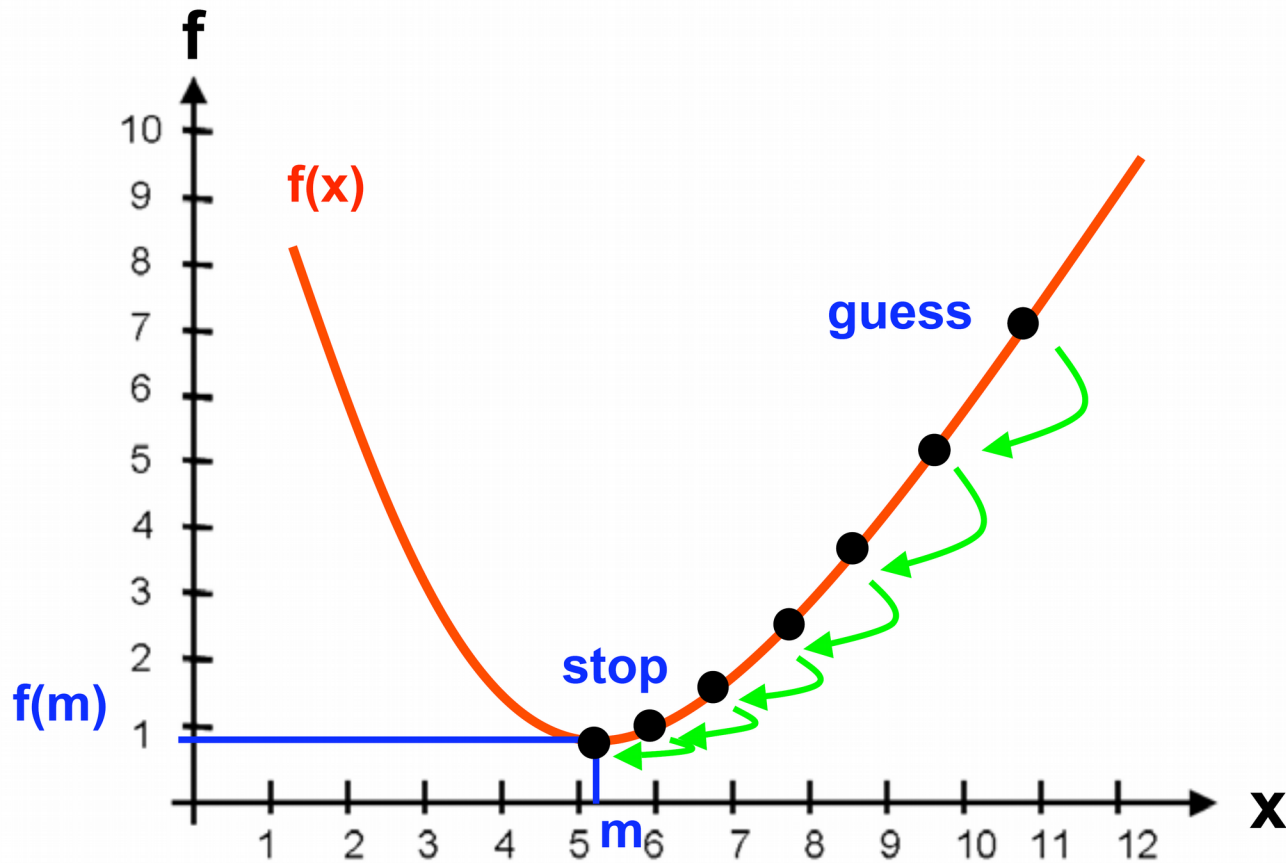
Loss function
measures how well classifier fits training data

Regularizer
prefers solutions that generalize well

$$\min_{\mathbf{w}, b} L(\mathbf{w}, b) = \min_{\mathbf{w}, b} \sum_{n=1}^N \mathbb{I}(y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

$\mathbb{I}(\cdot)$ Indicator function: 1 if (\cdot) is true, 0 otherwise
The loss function above is called the 0-1 loss

Illustrating gradient descent in 1-dimensional case



Gradient descent algorithm

Objective function
to minimize

Number of steps

Step size

Algorithm 22 GRADIENTDESCENT($\mathcal{F}, K, \eta_1, \dots$)

```
1:  $\mathbf{z}^{(0)} \leftarrow \langle 0, 0, \dots, 0 \rangle$  // initialize variable we are optimizing
2: for  $k = 1 \dots K$  do
3:    $\mathbf{g}^{(k)} \leftarrow \nabla_{\mathbf{z}} \mathcal{F}|_{\mathbf{z}^{(k-1)}}$  // compute gradient at current location
4:    $\mathbf{z}^{(k)} \leftarrow \mathbf{z}^{(k-1)} - \eta^{(k)} \mathbf{g}^{(k)}$  // take a step down the gradient
5: end for
6: return  $\mathbf{z}^{(K)}$ 
```

Recap: Linear Models

- General framework for binary classification
- Cast learning as optimization problem
- Optimization objective combines 2 terms
 - loss function: measures how well classifier fits training data
 - Regularizer: measures how simple classifier is
- Does not assume data is linearly separable
- Lets us separate model definition from training algorithm

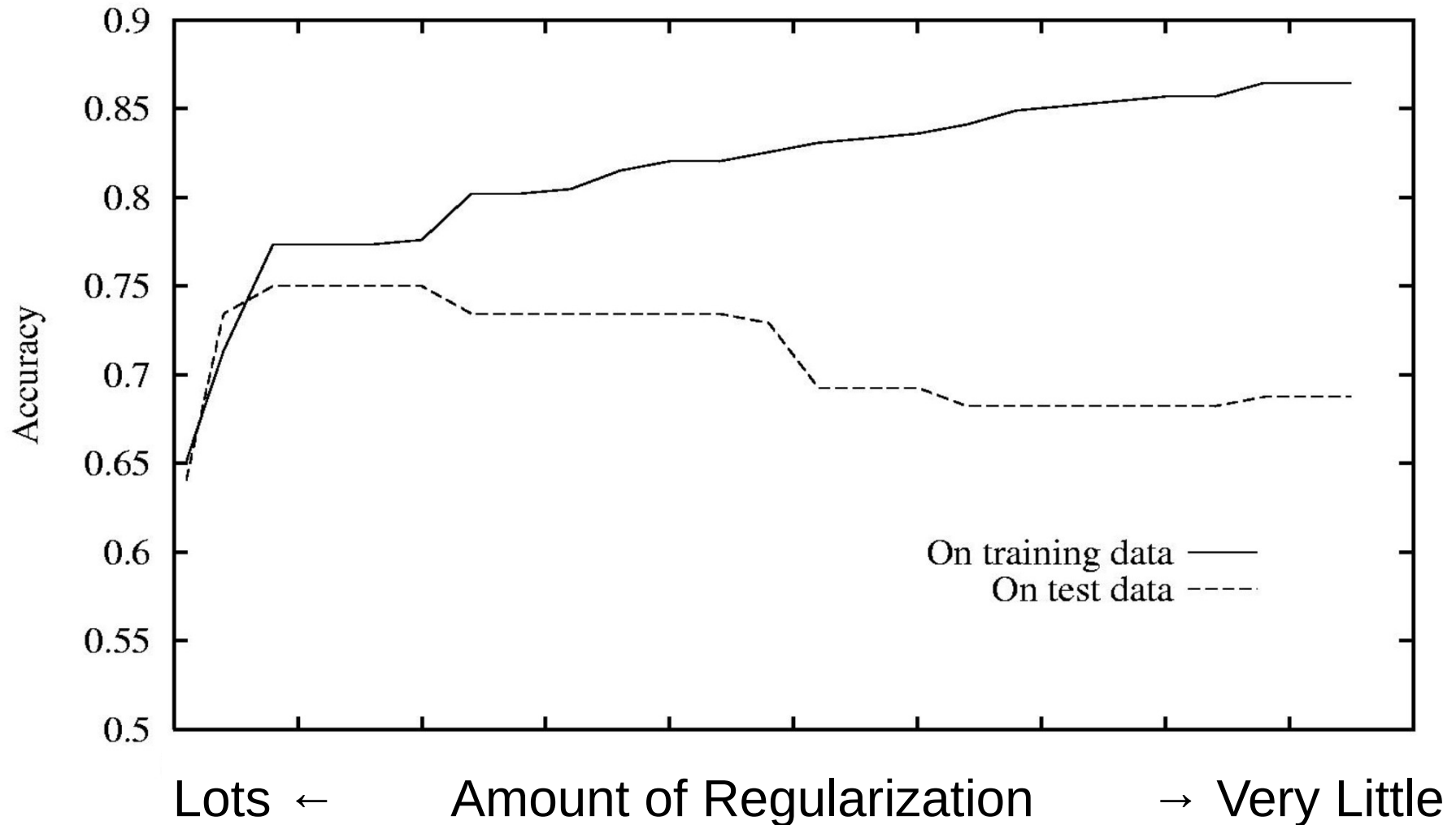
Overfitting

- Consider a hypothesis h and its:
 - Error rate over training data
 - True error rate over all data
- We say h overfits the training data if
Training error \ll Test error
- Amount of overfitting =
Test error – Training error

Evaluating on test data

- Problem: we don't know $error_{true}(h)$!
- Solution:
 - we set aside a test set
 - some examples that will be used for evaluation
 - we don't look at them during training!
 - after learning a model, called h , we calculate $error_{test}(h)$

Measuring effect of overfitting in linear models



Underfitting/Overfitting

- Underfitting
 - Learning algorithm had the opportunity to learn more from training data, but didn't
- Overfitting
 - Learning algorithm paid too much attention to idiosyncracies of the training data; the resulting tree doesn't generalize
- What we want:
 - A decision tree that neither underfits nor overfits
 - Because it is expected to do best in the future

Formalizing Errors

The learned classifier

\mathcal{F} set of all possible classifiers using a fixed representation

$$\text{error}(f) = \underbrace{\left[\text{error}(f) - \min_{f^* \in \mathcal{F}} \text{error}(f^*) \right]}_{\text{estimation error}} + \underbrace{\left[\min_{f^* \in \mathcal{F}} \text{error}(f) \right]}_{\text{approximation error}}$$

How far is the learned classifier f from the optimal classifier f^* ?

Quality of the model family
aka hypothesis class

The bias/variance trade-off

- Trade-off between
 - approximation error (bias)
 - estimation error (variance)
- Example:
 - Consider the always positive classifier
 - Low variance as a function of a random draw of the training set
 - Strongly biased toward predicting +1 no matter what the input

Ok, let's do a thought experiment...

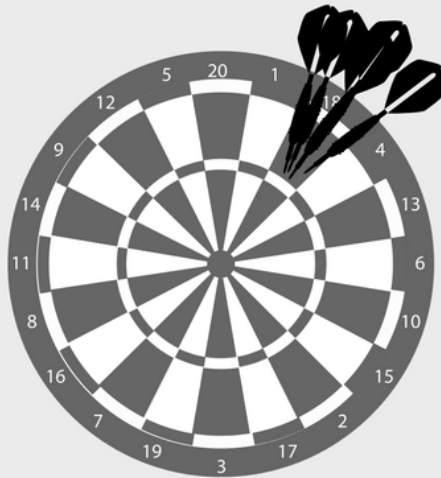
Imagine you've collected 5 different training datasets for the same problem. Now, imagine using one algorithm to train 5 models (one for each training set).



Here's what those 5 models tell you about your chosen algorithm:

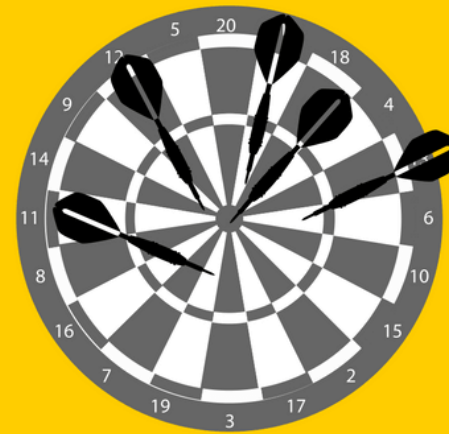


High Bias
Low Variance



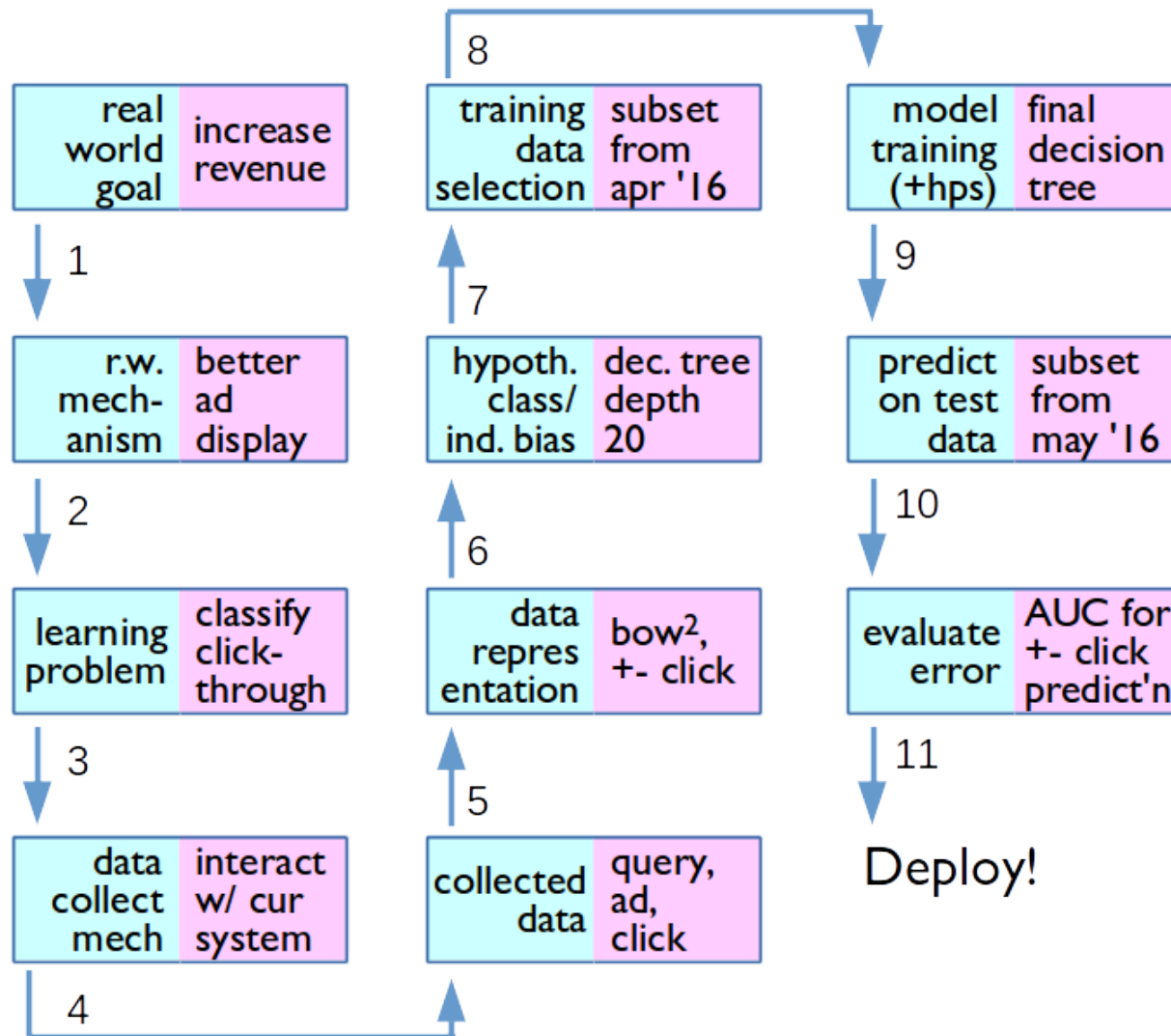
High bias, low variance algorithms train models that are consistent, but inaccurate *on average*.

High Variance
Low Bias



High variance, low bias algorithms train models that are accurate *on average*, but inconsistent.

Typical Design Process for an ML Application



Your homework assignments

- Let me know (go.umd.edu/mlvote):

- what your interests are
- what I'm doing well/poorly

- Pick some task (ideally a "social good" problem)

- reformulate as much of ML workflow as you can to that problem →
- (fig 2.4, ciml.info)

