CMU 15-781

Lecture 15:
Machine Learning I

Teacher:
Gianni A. Di Caro

Slides adapted from Ziko Kolter

OUTLINE

OUTLINE

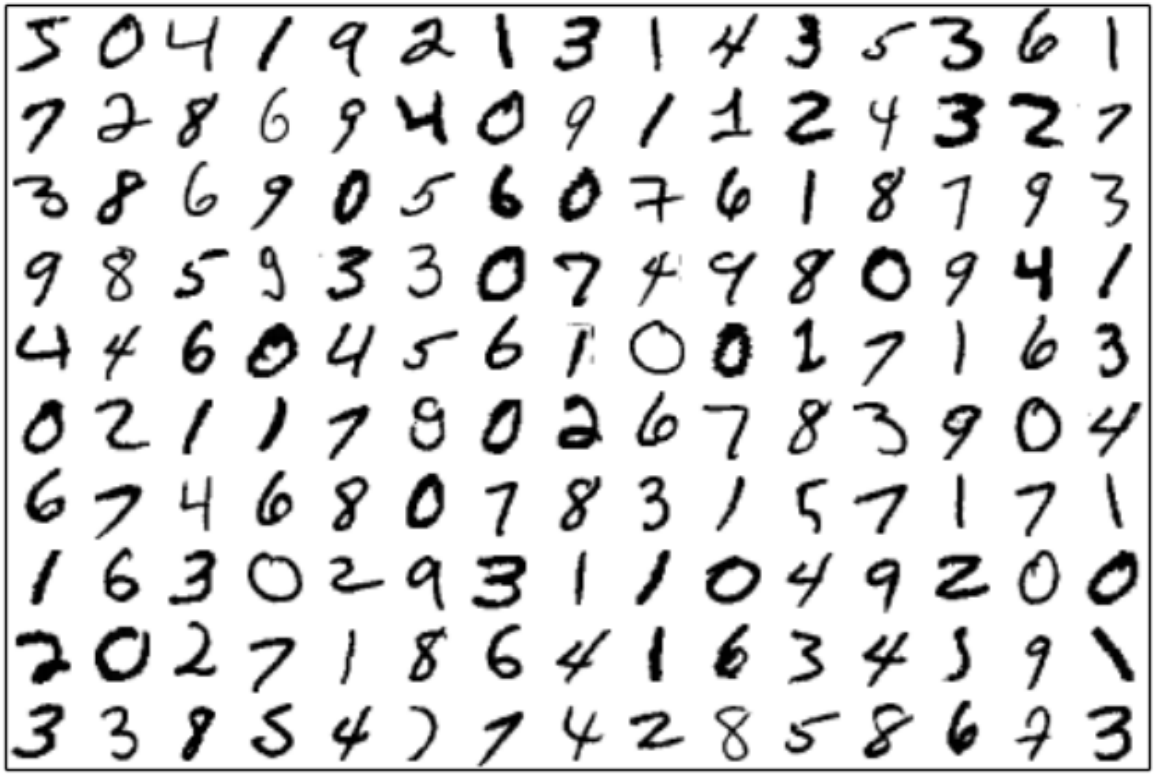
SOME DEFINITIONS

- **Improving performance via experience**
- Formally, A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, it its performance at tasks in T as measured by P, improves with experience (T. Mitchell)

- **The ability to perform a task in a situation which has never been encountered before (Learning $\rightarrow$ Generalization)**
- Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the task or tasks drawn from the same population more efficiently and more effectively the next time (H. Simon)

AN APPLICATION SCENARIO: DIGIT CLASSIFICATION

- The task: write a program that, given a 28x28 grayscale image of a digit, outputs the string representation



Digits from MNIST dataset (http://yann.lecun.com/exdb/mnist/)

POSSIBLE APPROACHES

- One approach, *direct modeling*: try to write a program by hand that uses your a priori knowledge of digits to properly classify the images

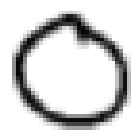
- Alternative method, *machine learning*: collect a bunch of images and their corresponding digits, write a program that uses this data to build its own method for classifying images

- (More precisely, this is a subset of machine learning called *supervised learning*)

A SUPERVISED LEARNING PIPELINE

Training Data

$\left(\text{  } , 2 \right)$

$\left(\text{  } , 0 \right)$

$\left(\text{  } , 8 \right)$

$\left(\text{  } , 5 \right)$

$\vdots$

Machine Learning

$\longrightarrow$ Hypothesis function h_θ

Deployment

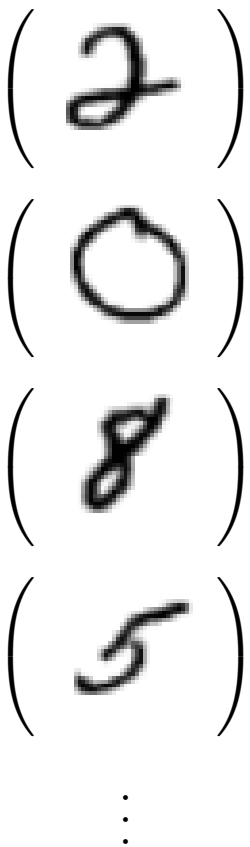
Prediction $= h_\theta \left(\text{  } \right)$

Prediction $= h_\theta \left(\text{  } \right)$

$\vdots$

UNSUPERVISED LEARNING

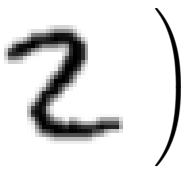
Training Data



Machine Learning

$\longrightarrow$ Hypothesis function h_θ

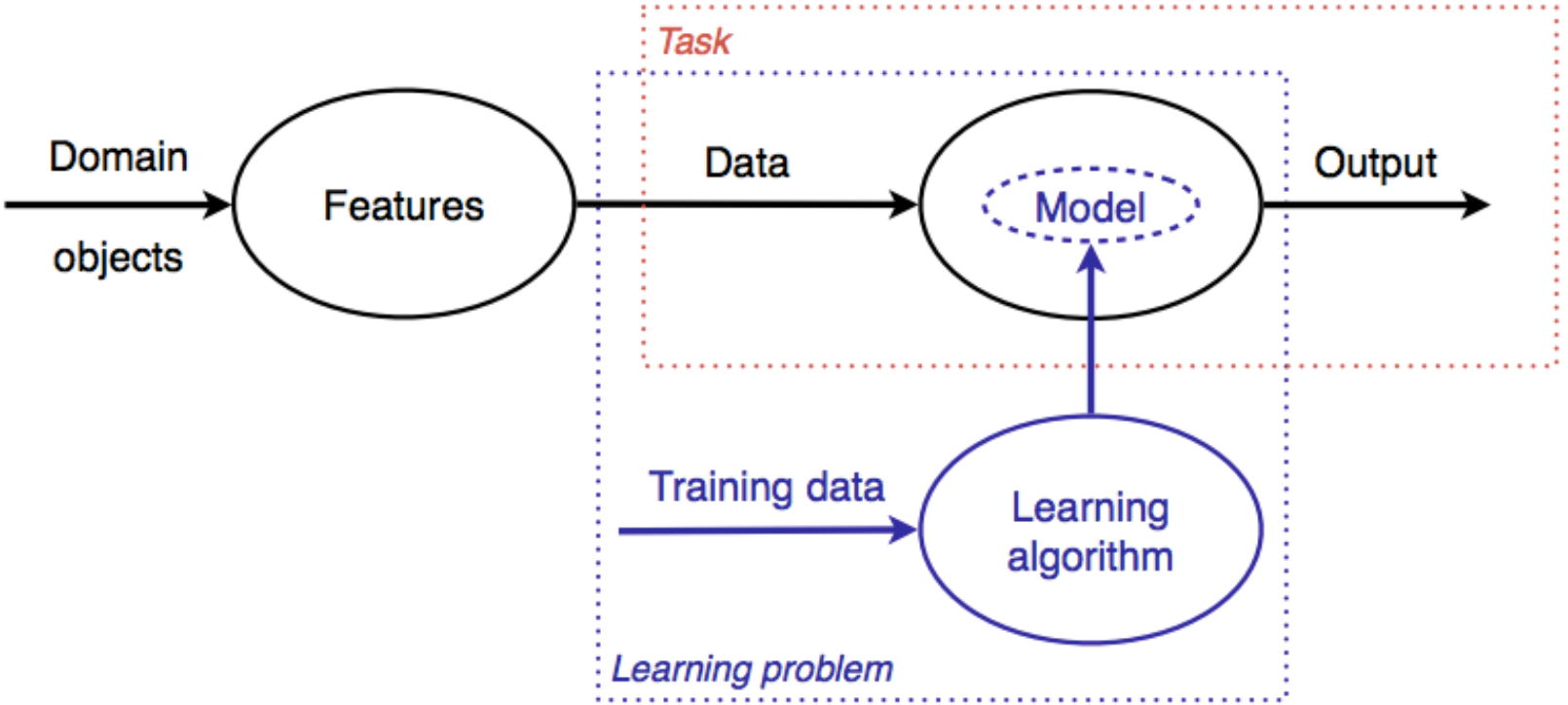
Deployment

Prediction $= h_\theta$ 

Prediction $= h_\theta$ 

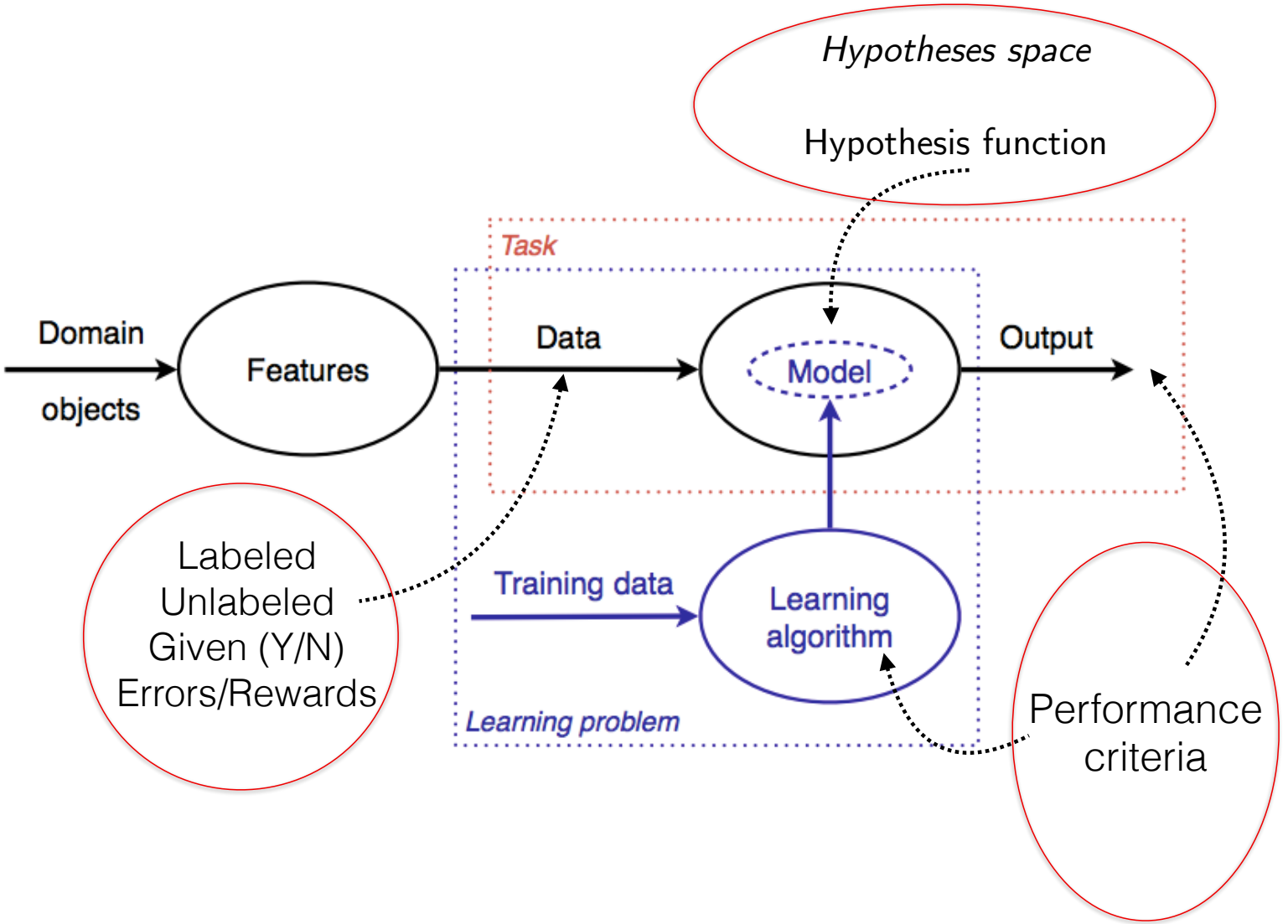
$\vdots$

GENERAL ML SCHEME



- A *Task* requires an appropriate **mapping** from data–described in the language of the features–to the outputs
- Obtaining such a mapping from **training data** is what constitutes a *Learning problem*
- *ML Design*: Use the right features (description language), to build the right model, that achieve the right task according to the desired performance

GENERAL ML SCHEME



TYPES OF LEARNING

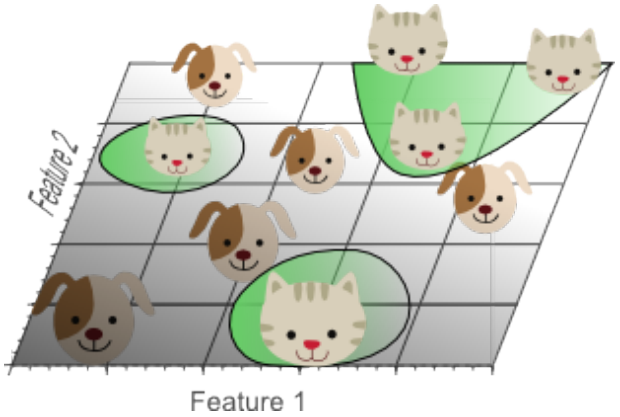
- **Supervised (inductive) learning** (*labeled data*)
  - A training data set is given
  - Training data include target outputs (put by a *teacher* / *supervisor*)
    $\rightarrow$ A precise error measure can be derived

- **Unsupervised learning** (*unlabeled data*)
  - A (training) data set is given
  - Training data does not include target outputs
  - Aims to find hidden structure, association relationships in data

- **Reinforcement learning** (*advisory signal*)
  - Explore to find the data by your own
  - Gathered data data does not include target outputs, but are associated to an advisory signal (reward)
  - Aims to learn an optimal action policy

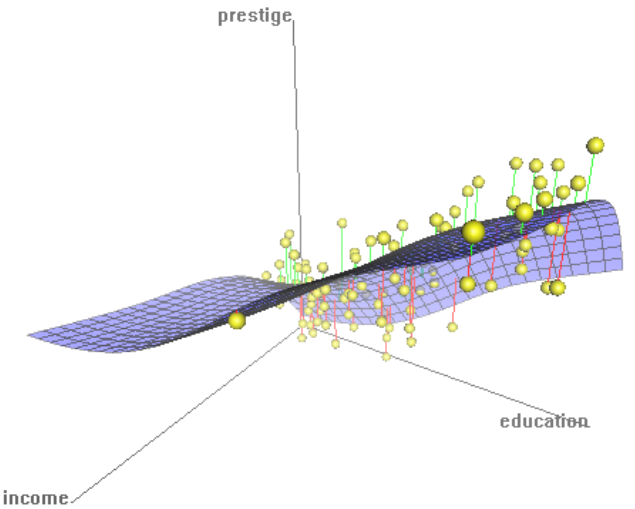
TYPES OF LEARNING TASKS

- **Classification, categorical target:** given n possible classes, to which class each data item belongs to?
  Task mapping is $t : \mathcal{F}^m \mapsto \mathbb{Z}^n$,
  - *Binary:* Apple or pear? 0 or 1? Slow or fast? Blue or red? Trick or treat?
  - *Multi-class:* Apple, or pear, or melon, or potato? 0, or 1, or 2, or . . . 1000? At rest, or very slow, or slow, or moderate, or fast, or very fast, or warp 1?



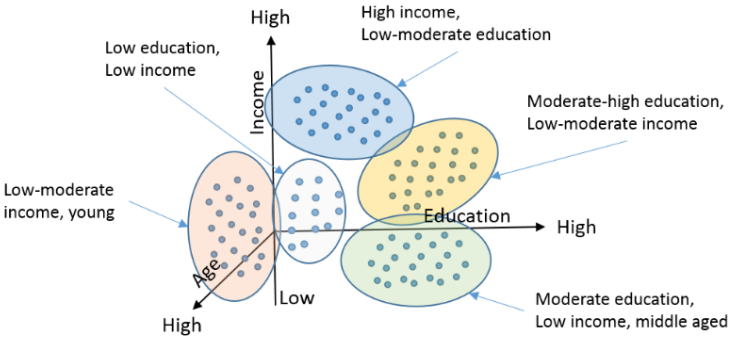
TYPES OF LEARNING TASKS

- **Regression, numerical target:** which is the function that best describes the (conditional expectation) relation between one or more dependent variables and one or more independent variables ("predictors")?
  Task mapping is $t : \mathcal{F}^m \mapsto \mathbb{R}^n$,
  - *Univariate:* What is the expected relation between temperature and peak electricity usage in Pittsburgh? What is the expected relation between age and height in China?
  - *Multi-variate:* What is the expected relation between (temperature, time hour, day of the week) and and peak electricity usage in Pittsburgh? What is the expected relation between (age, province, domestic income) and height in China?



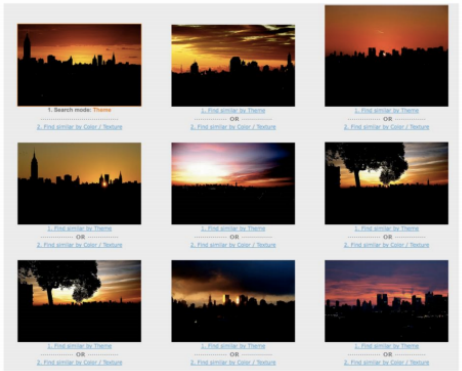
TYPES OF LEARNING TASKS

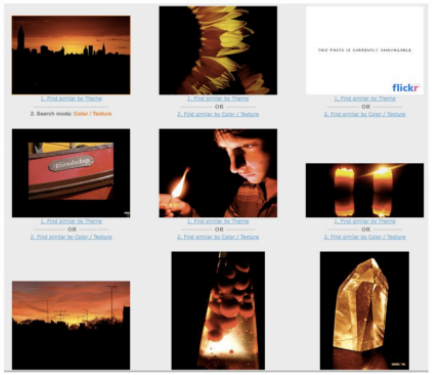
- **Clustering, hidden target:** based on some measure of similarity/dissimilarity, group data items in n clusters, n is (usually) not known in advance.
  Task mapping is $t : \mathcal{F}^m \mapsto \mathbb{Z}^n$,
  - Given a set of photos, and similarity features (e.g., colors, geometric objects) how can be the photos clustered? Given a set of students and their scores in the exams, how to partition them for recommending to work in different IT companies?



TYPES OF LEARNING TASKS

- **Finding underlying structure, hidden target:** discover relations and correlations among data.
  Task mapping is $t : \mathcal{F}^m \mapsto \mathbb{Z}^n$,
  - Given a set of photos, find possible relations among sub-groups of them. Given shopping data of customers at gas station spread in the country, discover relations that could suggest what to put in the shops and where to locate the items on display.





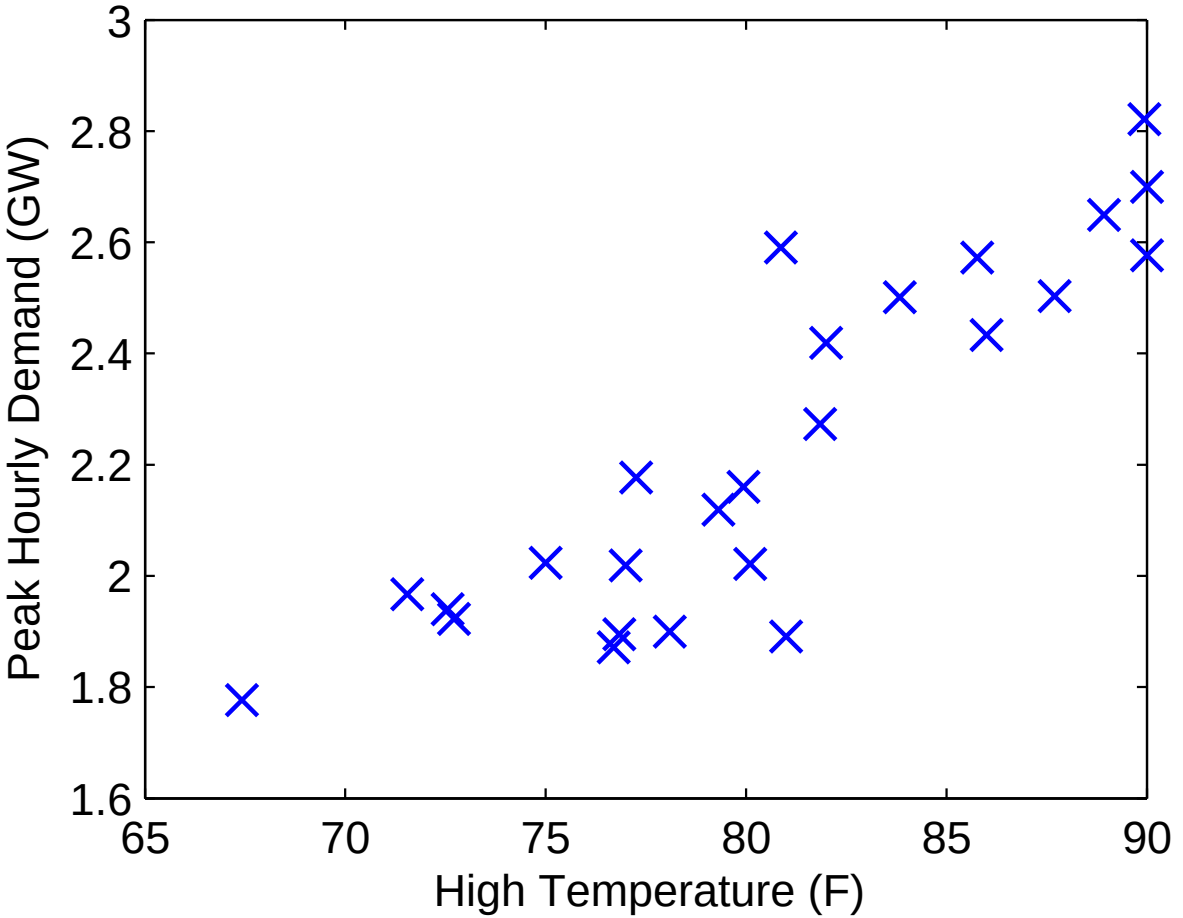
OUTLINE

A SIMPLE EXAMPLE: PREDICTING ELECTRICITY USE

- What will peak power consumption be in the Pittsburgh area tomorrow?
- Collect data of past high temperatures and peak demands

High Temperature (F)	**Peak Demand (GW)**
76.7	1.87
72.7	1.92
71.5	1.96
86.0	2.43
90.0	2.69
87.7	2.50
⋮	⋮

PREDICTING ELECTRICITY USE



Several days of peak demand vs. high temperature in Pittsburgh

PREDICTING ELECTRICITY USE: *LINEAR MODEL*

- **Hypothesize model**

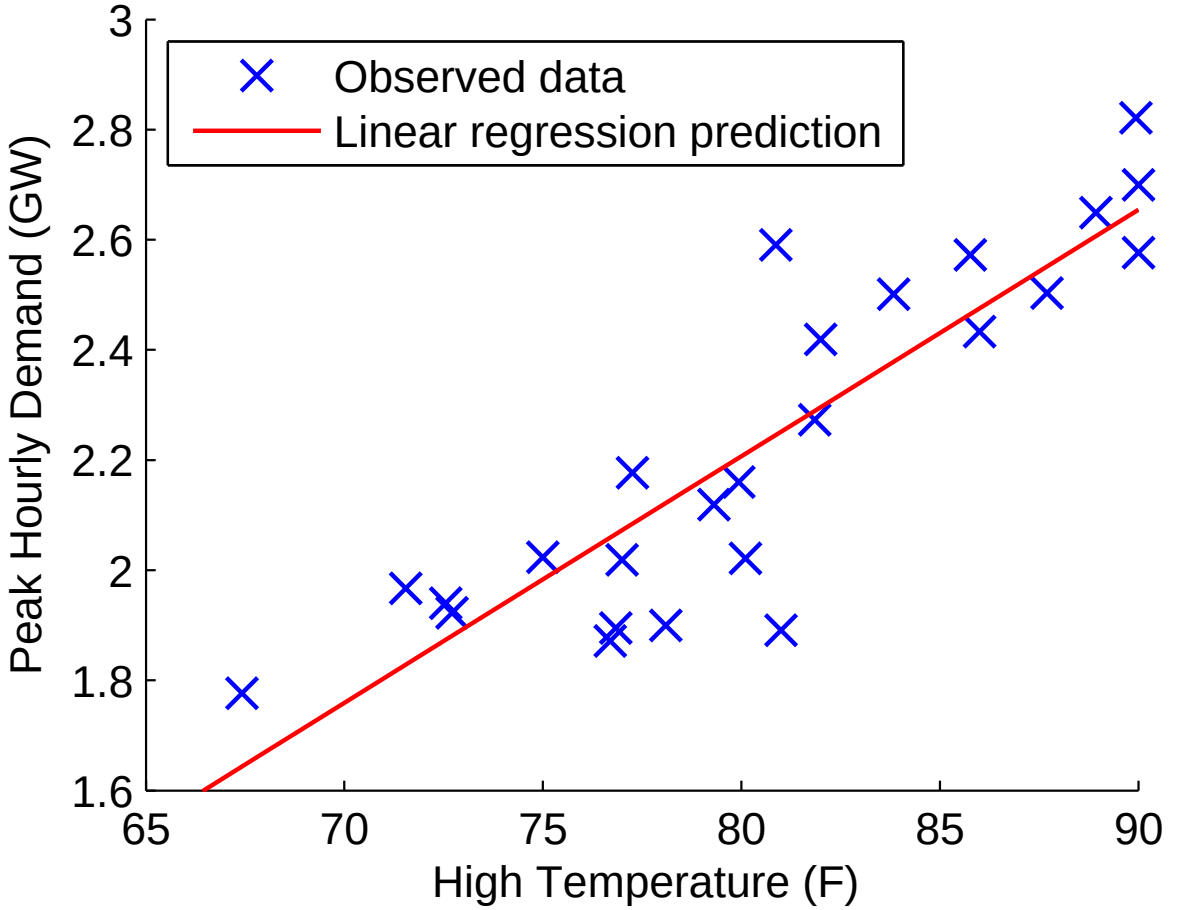
$$\text{Peak demand} \approx \theta_1 \cdot (\text{High temperature}) + \theta_2$$

  for some numbers θ_1 and θ_2

- Then, given a forecast of tomorrow's high temperature, we can *predict* the likely peak demand by plugging it into our model

LINEAR REGRESSION MODEL

- Equivalent to "drawing a line through the data"



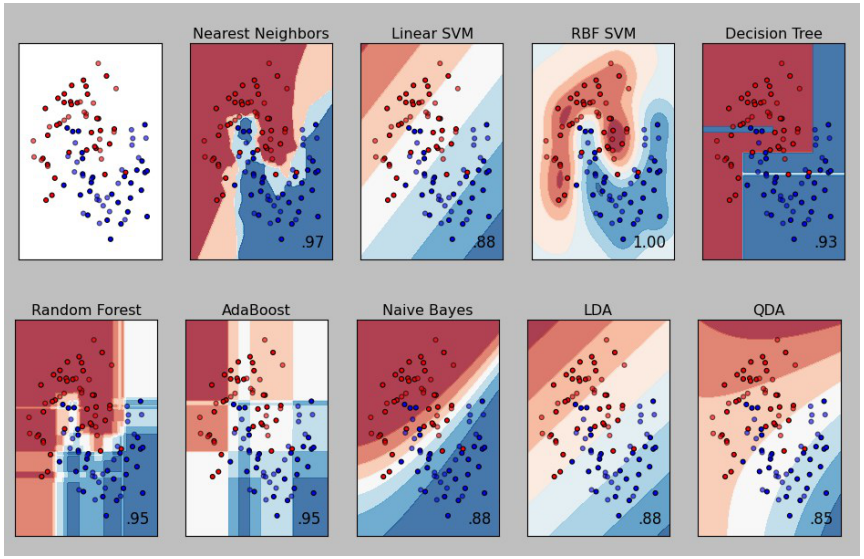
FORMAL LINEAR REGRESSION MODEL

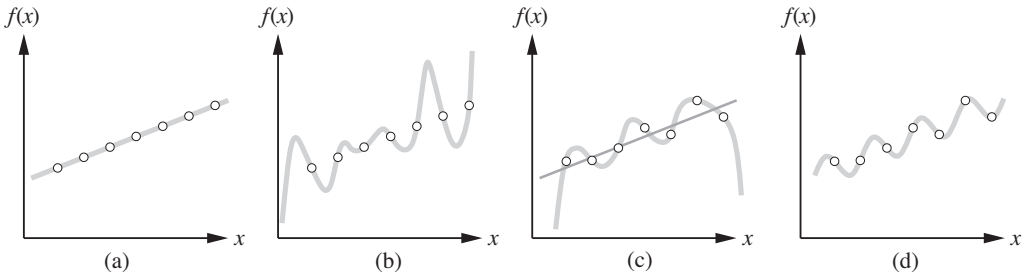
- **Input features**: $x^{(i)} \in \mathbb{R}^n, \quad i = 1, \ldots, m$
  - E.g.: $x^{(i)} \in \mathbb{R}^2 = \begin{bmatrix} \text{temperature for day } i \\ 1 \end{bmatrix}$
- **Output**: $y^{(i)} \in \mathbb{R}$ (*regression* task)
  - E.g.: $y^{(i)} \in \mathbb{R} = \{\text{peak demand for day } i\}$
- **Model Parameters**: $\theta \in \mathbb{R}^n$
- **Hypothesis function**: $h_\theta(x) : \mathbb{R}^n \to \mathbb{R}$
  - Hypothesis function: $h_\theta(x)$ returns a *prediction* of the output y, e.g. *linear regression*

$$h_\theta(x) = \boldsymbol{x}^T \boldsymbol{\theta} = \sum_{i=1}^{n} x_i \theta_i$$

Goal: Learn the (best) parameter vector $\boldsymbol{\theta}$ using the available training data $x^{(i)}, i = 1, \ldots, m$

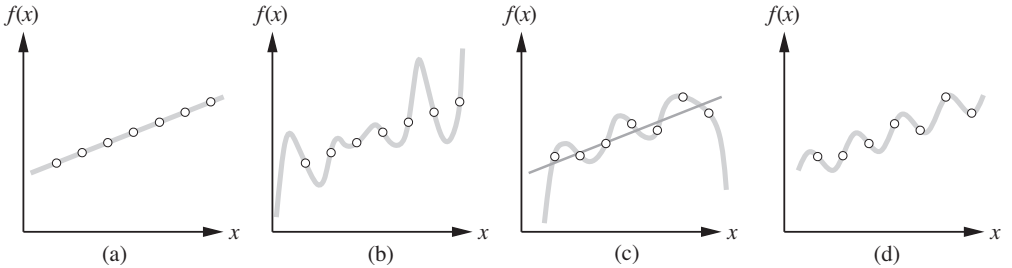
DO DESIGN CHOICES MATTER?





(PARAMETRIC) HYPOTHESIS SPACE

- Linear functions
- Polynomials of degree n
- Periodic functions
- ...



- A **consistent** hypothesis agrees will all data
- How do we choose among multiple consistent hypothesis? $\rightarrow$ Ockham's razor!
- Choose the *tradeoff* between complex hypotheses that fit the training data well and simpler hypotheses that may *generalize* better (e.g., the line in (c))
- A learning problem is **realizable** if the hypothesis space contains the true function (but we don't know it!)
- Tradeoff between **expressiveness** of a hypothesis space and **complexity** finding a good hypothesis within that space

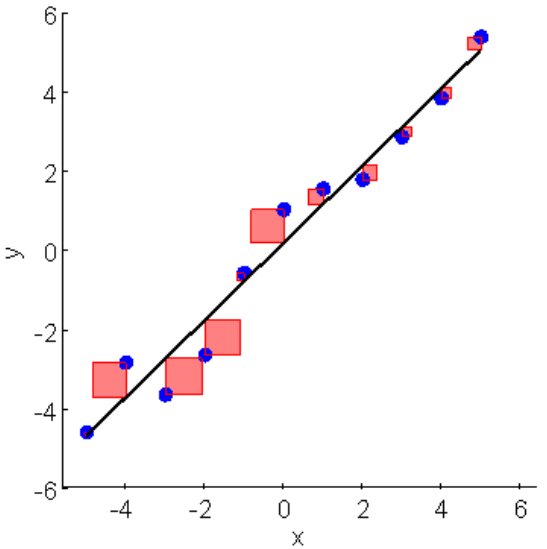
LOSS FUNCTIONS

- How do we measure how "good" a hypothesis is on the training data?
- Typically done by introducing a *loss function*

$$\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$$

- Intuitively, this function outputs a "small" value if $h_\theta(x)$ is "close" to y (the desired target output), a large value if it is "far" from y
- E.g., for regression, squared loss

$$\ell\left(h_\theta(x), y\right) = \left(h_\theta(x) - y\right)^2$$



THE *CANONICAL* ML PROBLEM

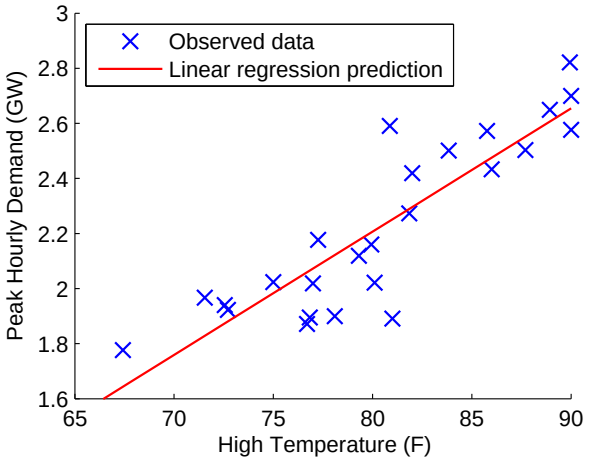
- Given a collection of input features and outputs $(x^{(i)}, y^{(i)})$, $i = 1, \ldots, m$, and a hypothesis function h_θ, find parameters θ that **minimize the sum of losses**

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{m} \ell\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

- Virtually all learning algorithms can be described in this form, we just need to specify three things:

  1. The hypothesis class: h_θ
  2. The loss function: ℓ
  3. The algorithm for solving the optimization problem (often approximately)

RETURN TO POWER DEMAND FORECASTING



- Linear hypothesis class: $h_\theta(x) = \boldsymbol{x}^T\boldsymbol{\theta}$
- Squared loss function: $\ell(h_\theta(y), y) = (h_\theta(\boldsymbol{x}) - y)^2$
- Resulting optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \ell\left(h_\theta(\boldsymbol{x}^{(i)}), y^{(i)}\right) \equiv \underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \left(\boldsymbol{x}^{(i)\,T}\boldsymbol{\theta} - y^{(i)}\right)^2$$

HOW DO WE SOLVE THE LINEAR REGRESSION PROBLEM?

- **Gradient descent** to solve optimization problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \left({x^{(i)}}^T \theta - y^{(i)} \right)^2$$

- Why GD?
- Gradient is given by

$$\begin{aligned} \nabla_\theta \sum_{i=1}^{m} \left({x^{(i)}}^T \theta - y^{(i)} \right)^2 &= \sum_{i=1}^{m} \nabla_\theta \left({x^{(i)}}^T \theta - y^{(i)} \right)^2 \\ &= 2 \sum_{i=1}^{m} x^{(i)} \left({x^{(i)}}^T \theta - y^{(i)} \right) \end{aligned}$$

- Gradient descent, repeat: $\theta \leftarrow \theta - \alpha \sum_{i=1}^{m} x^{(i)} \left({x^{(i)}}^T \theta - y^{(i)} \right)$

AN ANALYTIC SOLUTION IS ALSO AVAILABLE

- The function is convex, the point where the gradient is 0 is guaranteed to be the global minimum:

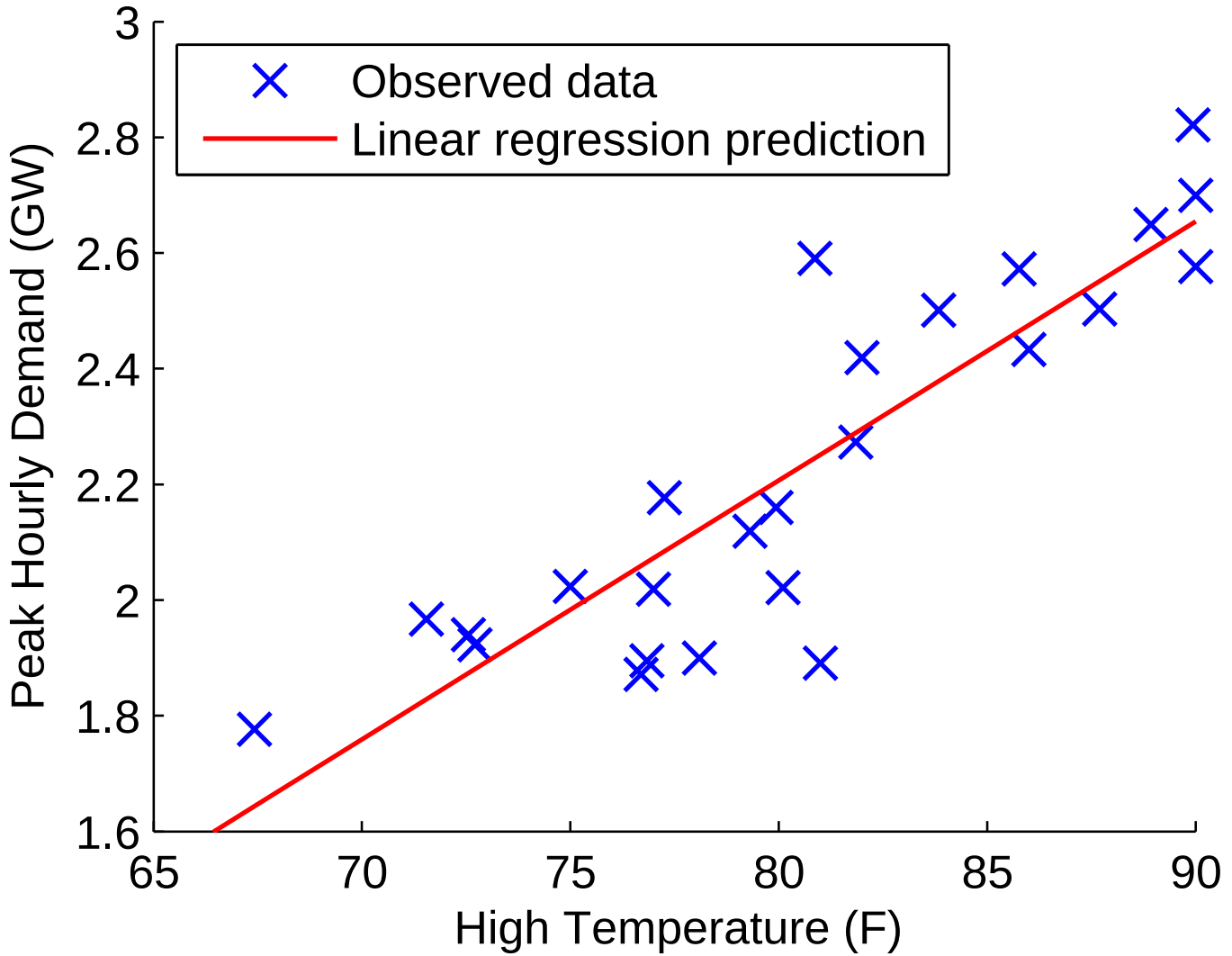
$$\nabla_\theta f(\theta) = \sum_{i=1}^{m} x^{(i)} \left({x^{(i)}}^T \theta^\star - y^{(i)} \right) = 0$$

$$\implies \left(\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T \right) \theta^\star = \sum_{i=1}^{m} x^{(i)} y^{(i)}$$

$$\implies \theta^\star = \left(\sum_{i=1}^{m} x^{(i)} {x^{(i)}}^T \right)^{-1} \left(\sum_{i=1}^{m} x^{(i)} y^{(i)} \right)$$

- Squared loss is one of the few cases that such directly solutions are possible, usually need to resort to gradient descent or other methods

THE RESULT FOR THE POWER DEMAND



GRADIENT DESCENT FOR THE GENERAL ML PROBLEM

- GD can be used for the generic optimization problem in ML:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \ell(h_\theta(x^{(i)}), y^{(i)})$$

- In the case of non convex functions, only a local minimum can be guaranteed
- Procedurally, gradient descent then takes the form:

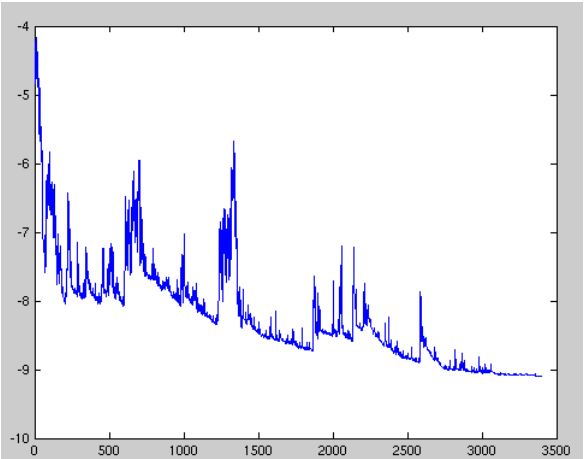
function $\theta = \text{Gradient-Descent}(\{(x^{(i)}, y^{(i)})\}, h_\theta, \ell, \alpha)$
    Initialize: $\theta \leftarrow 0$
    Repeat until convergence
        $g \leftarrow 0$
        For $i = 1, \ldots, m$:
            $g \leftarrow g + \nabla_\theta \ell(h_\theta(x^{(i)}), y^{(i)})$
        $\theta \leftarrow \theta - \alpha g$
    return θ

LARGE PROBLEMS: STOCHASTIC GRADIENT DESCENT

- If the number of samples m is large, computing a *single* gradient step is costly
- An alternative approach, **stochastic gradient descent (SGD)**, update the parameters based upon gradient *each* sample:

function $\theta = \text{SGD}(\{(x^{(i)}, y^{(i)})\}, h_\theta, \ell, \alpha)$
  Initialize: $\theta \leftarrow 0$
  Repeat until convergence
    For $i = 1, \ldots, m$ (randomly shuffle the order):
      $\theta \leftarrow \theta - \alpha \nabla_\theta \ell(h_\theta(x^{(i)}), y^{(i)})$
  return θ

- Can be viewed as taking many more steps along *noisy* estimates of the gradient, and often converges to a "good" parameter value after relatively few passes over the data set



The value of the function does not decrease monotonically

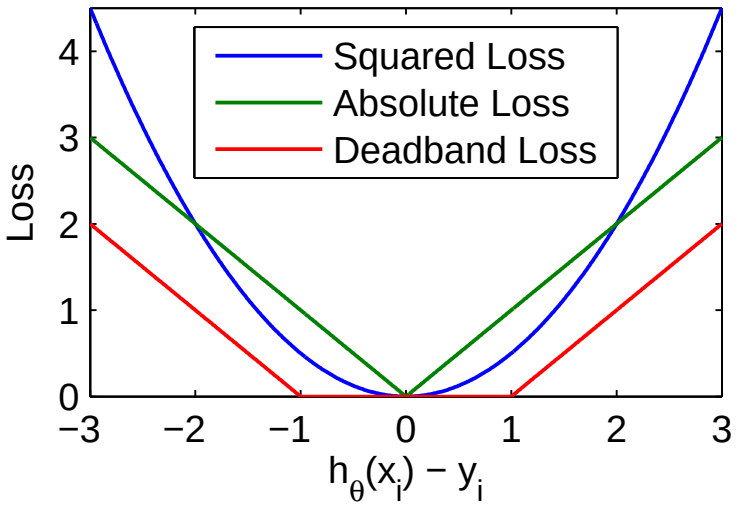
ALTERNATIVE LOSS FUNCTIONS

- Why did we choose the squared loss function

$$\ell\left(h_\theta(x), y\right) = \left(h_\theta(x) - y\right)^2?$$

- Some other alternatives

$$\begin{aligned} \text{Absolute loss:} \quad & \ell(h_\theta(x), y) = |h_\theta(x) - y| \\ \text{Deadband loss:} \quad & \ell(h_\theta(x), y) = \max\{0, |h_\theta(x) - y| - \epsilon\}, \ \epsilon \in \mathbb{R}_+ \end{aligned}$$



- A deviation from the target is penalized in different measure, which has an impact both on computation and on the quality of the final result

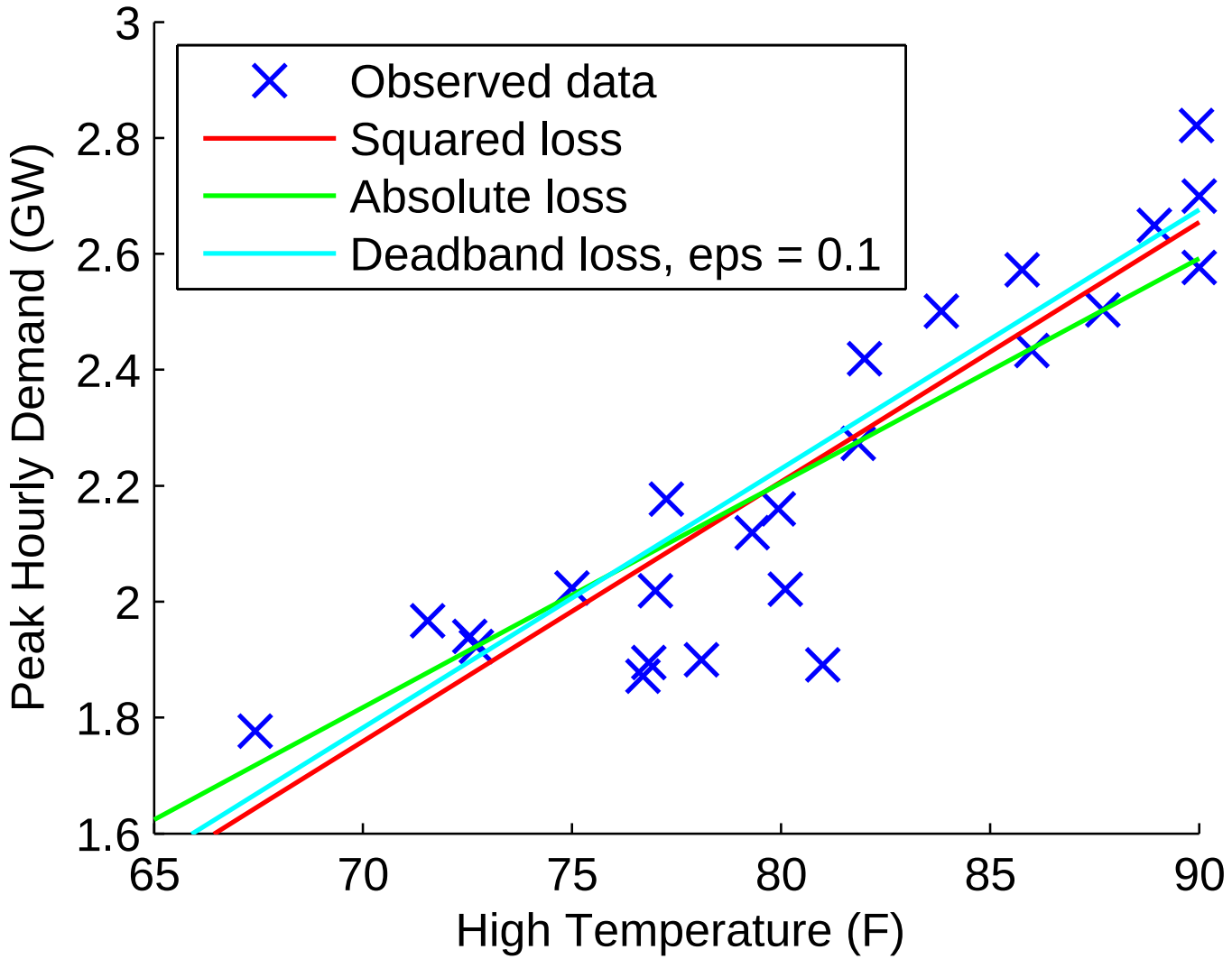
ALTERNATIVE LOSS FUNCTIONS

- For these loss functions, no closed-form expression for $\theta^\star$, but (sub)gradient descent can still be very effective
- E.g., for absolute loss and linear hypothesis class

$$\text{Repeat}: \theta \leftarrow \theta - \alpha \sum_{i=1}^{m} x^{(i)} \text{sign}\left({x^{(i)}}^T \theta - y^{(i)} \right)$$

- Can also solve for non-smooth losses using constrained optimization (and libraries like cvxpy)

EFFECT IN POWER PROBLEM



PROBABILISTIC INTERPRETATION

- Suppose that *each output* y in our data really *is* equal to the hypothesis function for that example, $h_\theta(x)$, just corrupted by **Gaussian noise** ϵ

$$y = h_\theta(x) + \epsilon$$

- The probability density of a Gaussian variable given by

$$p(\epsilon) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

- Substituting terms, we can use this expression to write the probability of the output y *given* the input data features x (parametrized by θ)

$$p(y|x;\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(h_\theta(x) - y)^2}{2\sigma^2}\right)$$

PROBABILISTIC INTERPRETATION

- Consider the joint probability of *all* training data, and let's assume that samples are independent and identically distributed:

$$p(y^{(1)}, \ldots, y^{(m)} | x^{(1)}, \ldots, x^{(m)}; \theta) = \prod_{i=1}^{m} p(y^{(i)} | x^{(i)}; \theta)$$

  where each $p(y^{(i)}|x^{(i)};\theta)$ is a Gaussian posterior

- Let's find the parameters θ that maximize the probability of the observed outputs given the input data (i.e., let's choose the hypothesis which is the most probable given the data):

$$\begin{aligned}
&\underset{\theta}{\text{maximize}} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \quad \equiv \quad \underset{\theta}{\text{minimize}} -\sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) \\
&\equiv \underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \left(\log(\sqrt{2\pi}\sigma) + \frac{1}{2\sigma^2}(h_\theta(x^{(i)}) - y^{(i)})^2 \right) \\
&\equiv \underset{\theta}{\text{minimize}} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2
\end{aligned}$$

PROBABILISTIC INTERPRETATION

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

But this is the same as the loss minimization problem!

- This is a procedure known as **maximum likelihood estimation**, a common statistical technique
- Note that we still just pushed the question of "which loss" to "which distribution"
  - But some distributions, like Gaussian, may have reasonable empirical or theoretical justifications for certain problems

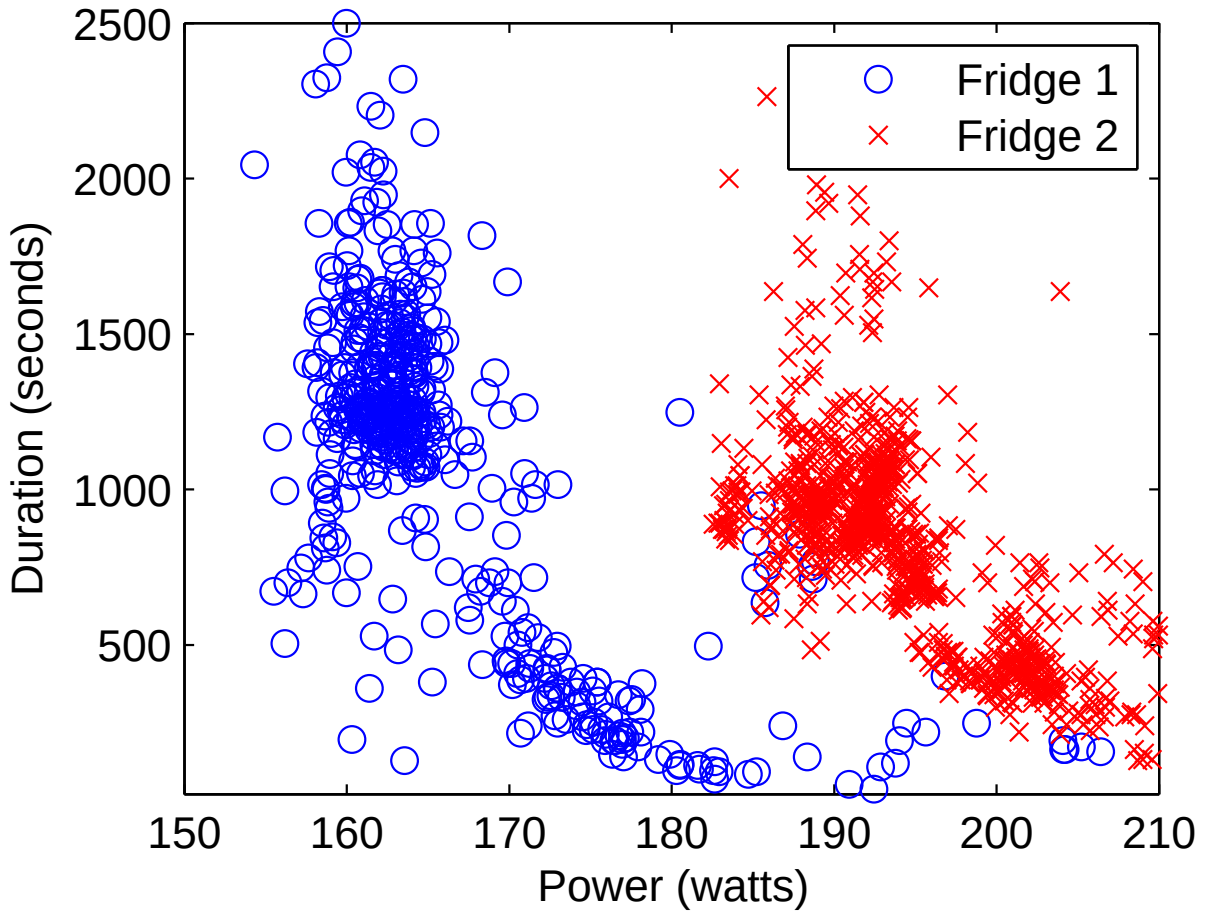
OUTLINE

CLASSIFICATION PROBLEMS

- Sometimes we want to predict discrete outputs rather than continuous
- Is the email spam or not? (YES/NO)
- What digit is in this image? (0/1/2/3/4/5/6/7/8/9)

EXAMPLE: CLASSIFYING HOUSEHOLD APPLIANCES

- Differentiate between two refrigerators using their power consumption signatures

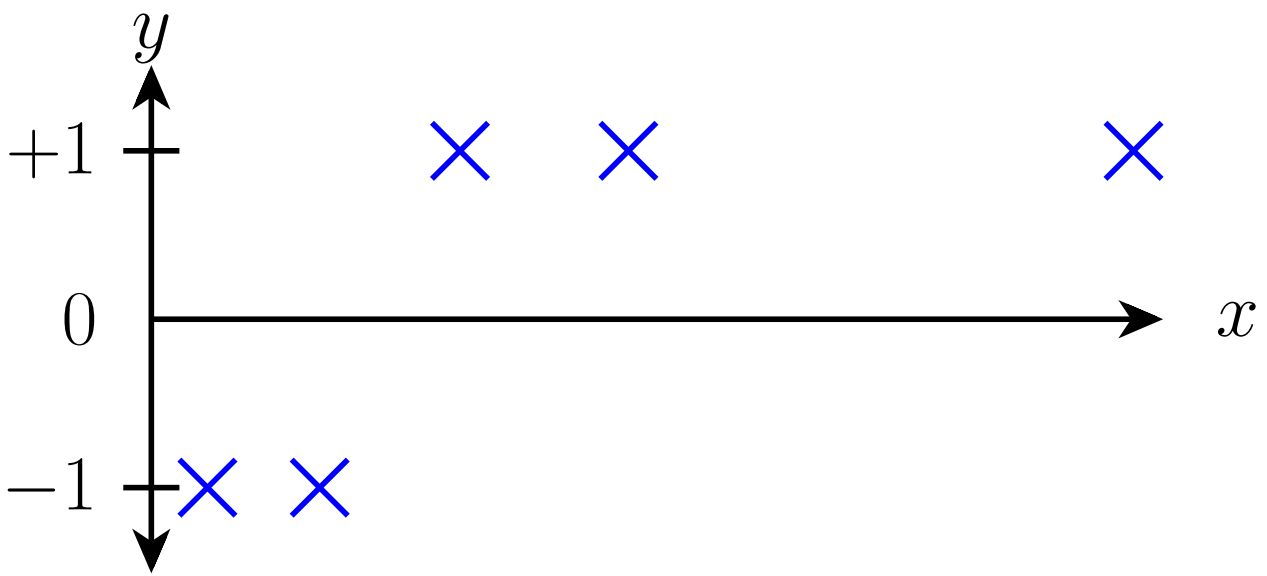


CLASSIFICATION TASKS

- **Input features**: $x^{(i)} \in \mathbb{R}^n, \quad i = 1, \ldots, m$
  - E.g.: $x^{(i)} \in \mathbb{R}^3 = (\text{Duration } i, \text{Power } i, 1)$
- **Output**: $y^{(i)} \in \{-1, +1\}$ (binary classification task)
  - E.g.: $y^{(i)} = \text{Is it fridge 1?}$
- **Model Parameters**: $\theta \in \mathbb{R}^n$
- **Hypothesis function**: $h_\theta(x) : \mathbb{R}^n \to \mathbb{R}$
  - Returns *continuous* prediction of the output y, where the value indicates how "confident" we are that the example is -1 or $+1$; $\text{sign}(h_\theta(x))$ is the actual binary prediction
  - Again, we will focus initially on *linear predictors* $h_\theta(x) = x^T\theta$

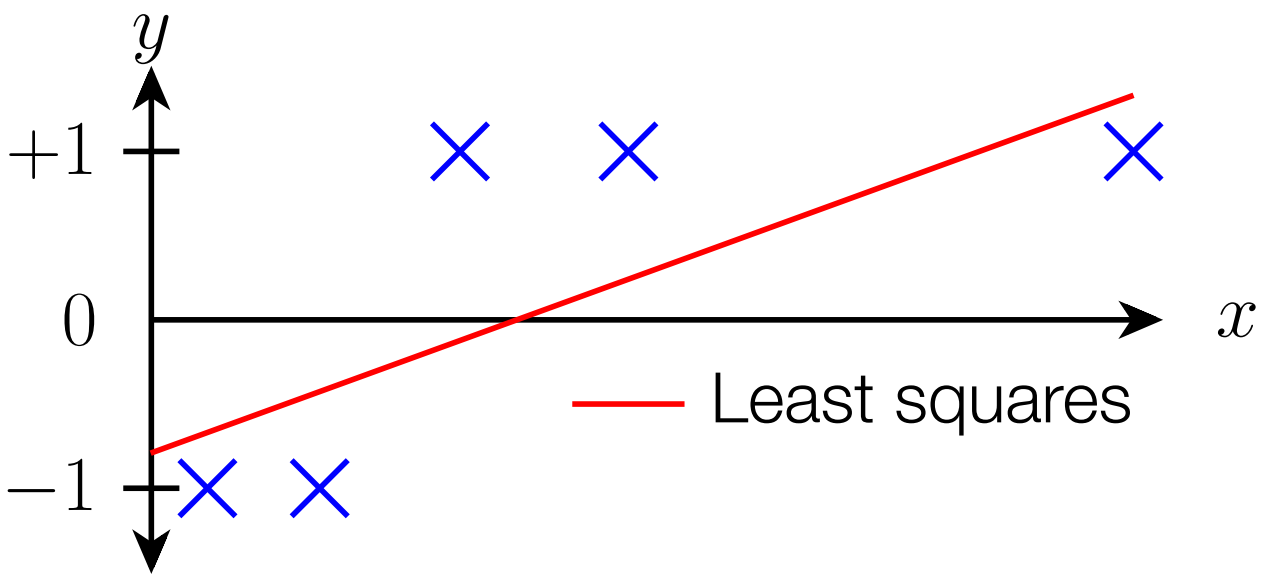
LOSS FUNCTIONS

- Loss function $\ell : \mathbb{R} \times \{-1, +1\} \to \mathbb{R}_+$



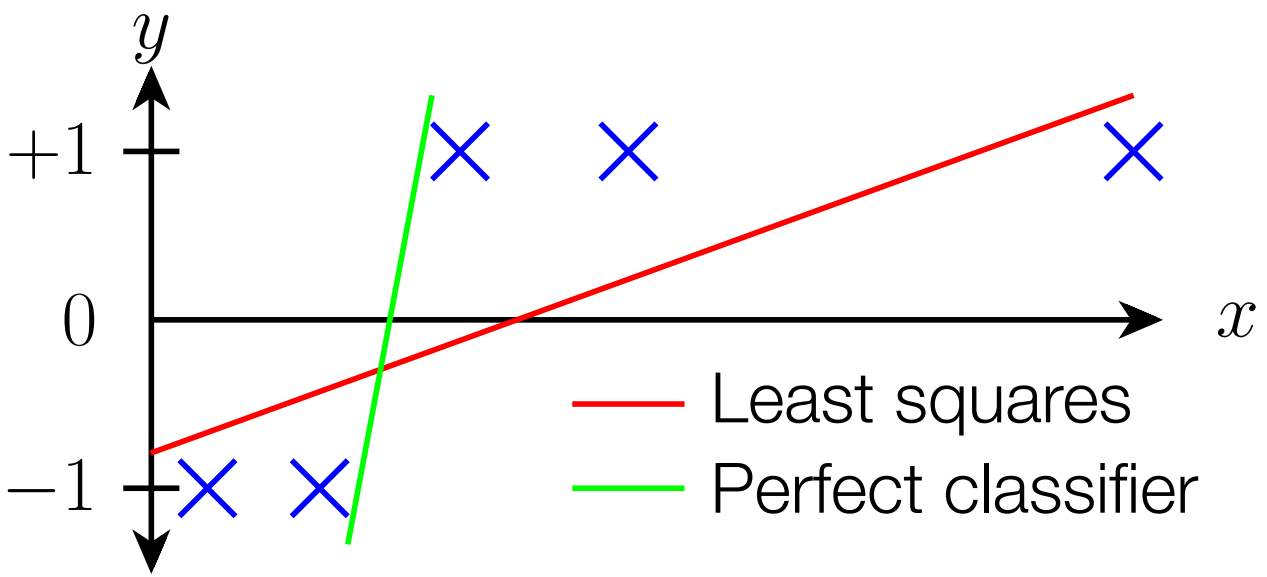
LOSS FUNCTIONS

- Loss function $\ell : \mathbb{R} \times \{-1, +1\} \to \mathbb{R}_+$
- Do we need a different loss function?



LOSS FUNCTIONS

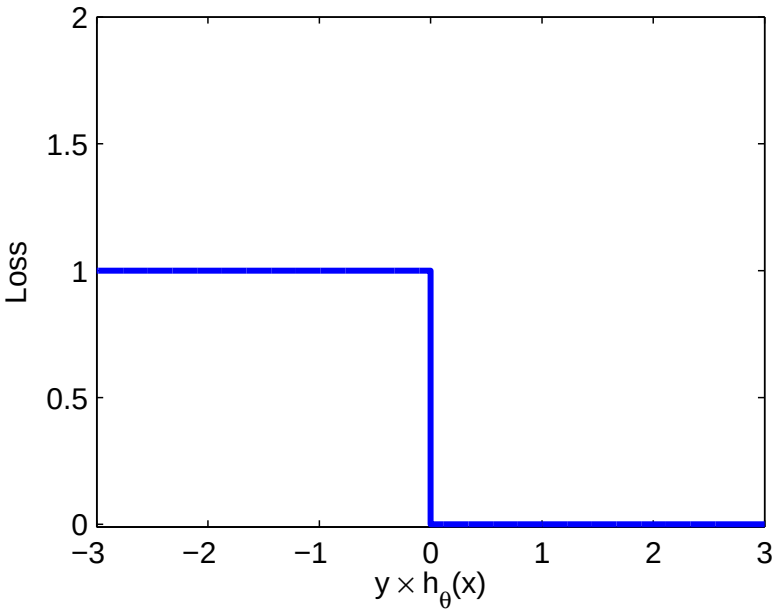
- Loss function $\ell : \mathbb{R} \times \{-1, +1\} \to \mathbb{R}_+$
- Do we need a different loss function?



ACCURACY LOSS FUNCTION

- The simplest loss (0/1 loss, **accuracy**): count the number of mistakes we make

$$\ell(h_\theta(x), y) = \begin{cases} 1 & \text{if } y \neq \text{sign}(h_\theta(x)) \\ 0 & \text{otherwise} \end{cases}$$
$$= \mathbf{1}\{y \cdot h_\theta(x) \leq 0\}$$

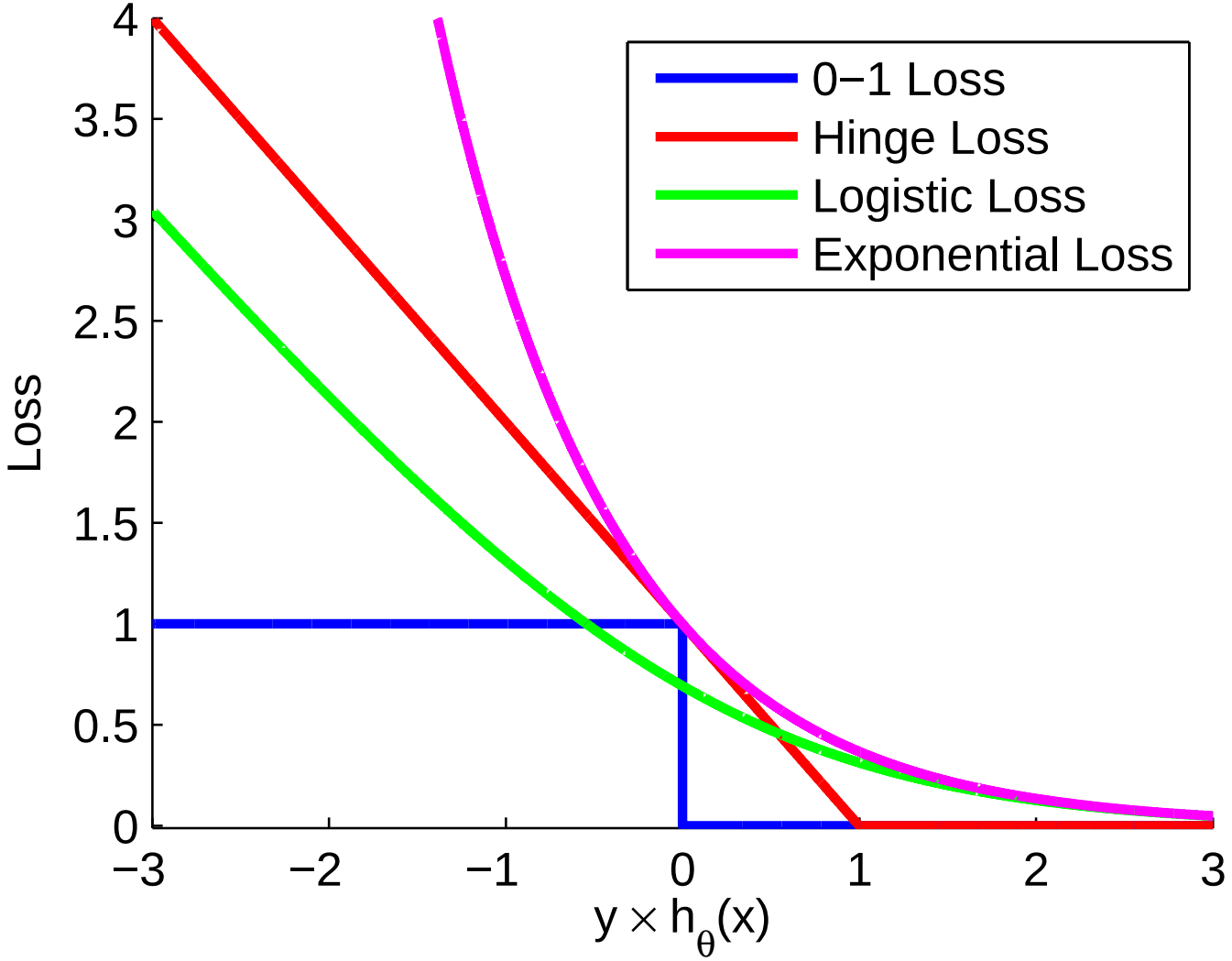


$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{m} \ell\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

SMOOTHING ACCURACY LOSS

- Unfortunately, minimizing sum of 0/1 losses leads to a hard optimization problem
- Because of this, a whole range of alternative "approximations" to 0/1 loss are used instead

$$\begin{aligned}
\text{Hinge loss:} \quad & \ell(h_\theta(x), y) = \max\{1 - y \cdot h_\theta(x), 0\} \\
\text{Squared hinge loss:} \quad & \ell(h_\theta(x), y) = \max\{1 - y \cdot h_\theta(x), 0\}^2 \\
\text{Logistic loss:} \quad & \ell(h_\theta(x), y) = \log(1 + e^{-y \cdot h_\theta(x)}) \\
\text{Exponential loss:} \quad & \ell(h_\theta(x), y) = e^{-y \cdot h_\theta(x)}
\end{aligned}$$



Common loss functions for classification