# Artificial Intelligence II (CS4442B & CS9542B)

## Supervised Learning: Linear Regression

Boyu Wang
Department of Computer Science
University of Western Ontario

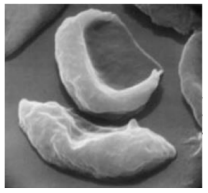
# Motivation examples

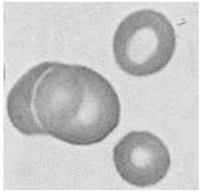
**Classification**

X = Document

Sports
Science
News

Y = Topic





X = Cell Image

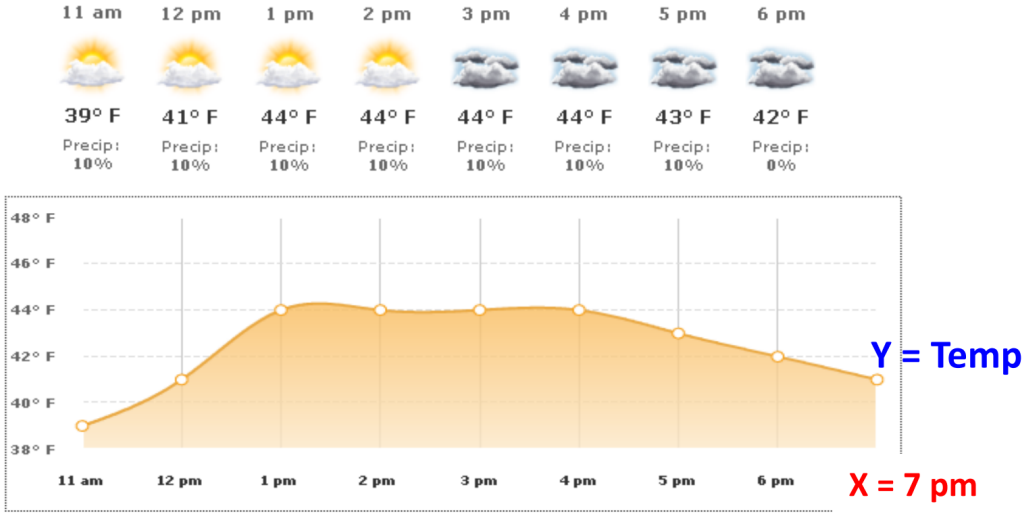
Anemic cell
Healthy cell

Y = Diagnosis

Slide credit: Maria-Florina Balcan

# Motivation examples

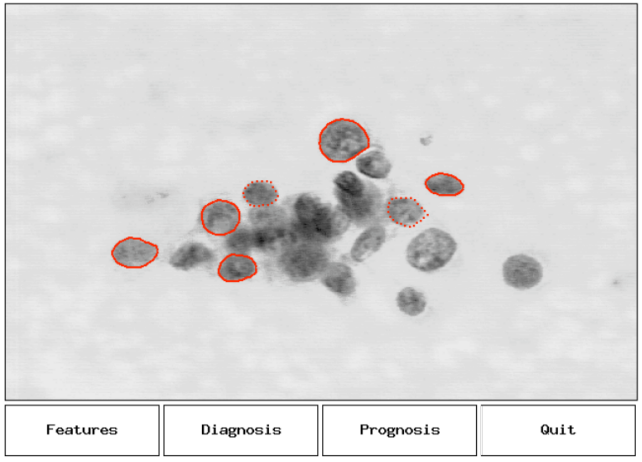
Weather Prediction




Estimating Contamination


Slide credit: Maria-Florina Balcan

# Motivation examples




- Cell samples were taken from tumors in breast cancer patients before surgery, and imaged
- Tumors were excised
- Patients were followed to determine whether or not the cancer recurred, and how long until recurrence or disease free

Slide credit: Doina Precup

# Data

- Thirty real-valued variables per tumor.
- Two variables that can be predicted:
  - Outcome (R=recurrence, N=non-recurrence)
  - Time (until recurrence, for R, time healthy, for N).

| tumor size | texture | perimeter | . . . | outcome | time |
|---|---|---|---|---|---|
| 18.02 | 27.6 | 117.5 | | N | 31 |
| 17.99 | 10.38 | 122.8 | | N | 61 |
| 20.29 | 14.34 | 135.1 | | R | 27 |
| . . . | | | | | |

Slide credit: Doina Precup

# Terminology

| tumor size | texture | perimeter | . . . | outcome | time |
|---|---|---|---|---|---|
| 18.02 | 27.6 | 117.5 | | N | 31 |
| 17.99 | 10.38 | 122.8 | | N | 61 |
| 20.29 | 14.34 | 135.1 | | R | 27 |
| . . . | | | | | |

- Columns are called features or attributes or input variables.
- The outcome and time (which we are trying to predict) are called output variables or targets.
- A row in the table is called training example or instance.
- The problem of predicting the recurrence is called (binary) classification.
- The problem of predicting the time is called regression.

# Problem formulation

| tumor size | texture | perimeter | . . . | outcome | time |
|---|---|---|---|---|---|
| 18.02 | 27.6 | 117.5 | | N | 31 |
| 17.99 | 10.38 | 122.8 | | N | 61 |
| 20.29 | 14.34 | 135.1 | | R | 27 |
| . . . | | | | | |

- A training example $i$ has the form: $(x_i, y_i)$, where $x_i \in \mathbb{R}^n$ is the number of features (feature dimension), and $y_i \in \{0, 1\}$ for (binary) classification, and $y_i \in \mathbb{R}$ for regression.
- The training set consists of $m$ examples: $S = \{(x_i, y_i)\}_{i=1}^m$.
- We denote the matrix of features by $X = [x_1, \ldots, x_m]^\top \in \mathbb{R}^{m \times n}$, and the column vector of outputs by $Y \in \mathbb{R}^m$.
- Objective: construct a good predictor using training data $S$ (i.e., $X$ and $Y$), such that it works well for new patients (test data).

# Problem formulation

- Let $\mathcal{X}$ denote the space of input values
- Let $\mathcal{Y}$ denote the space of output values
- Given a data set $S \in \mathcal{X} \times \mathcal{Y}$, find a function:

$$h : \mathcal{X} \to \mathcal{Y}$$

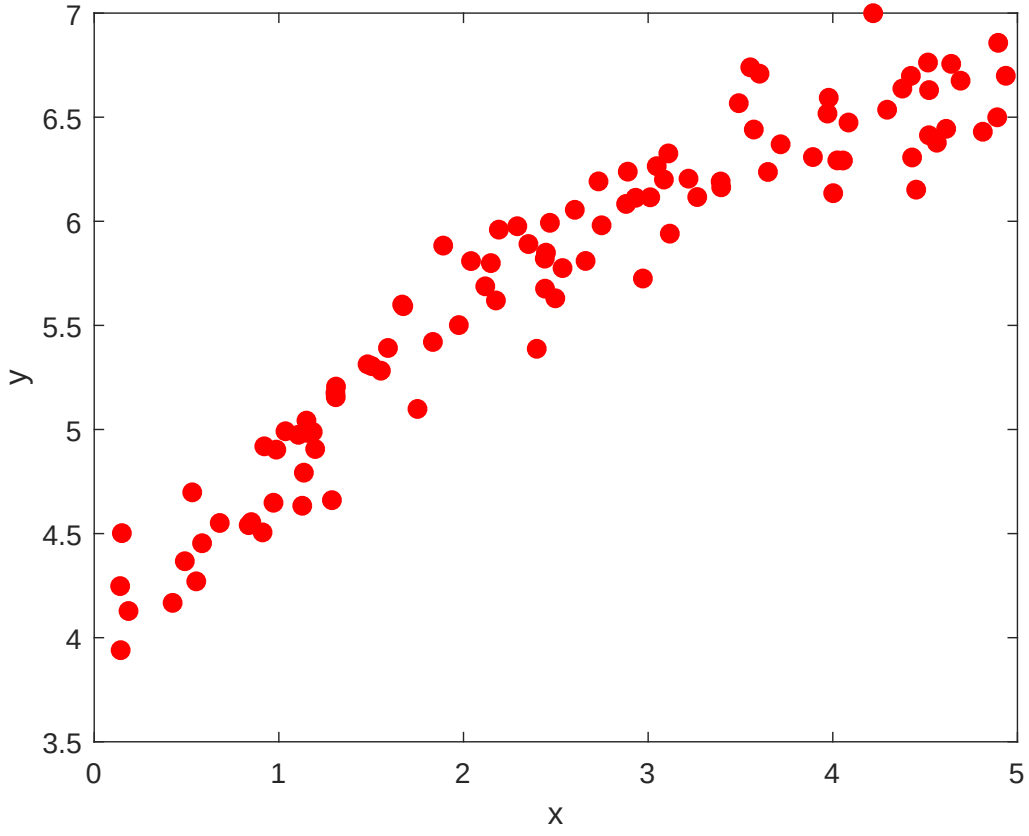
  such that for a new example $(x, y)$, $h(x)$ can correctly predict the value of $y$.

- Types of supervised learning problem: classification, regression, ranking, structured prediction...
- Key assumption: training and test data are sampled from the same distribution.

# Steps to solving a supervised learning problem

- Collect the data set (input-output pairs)
- Choose a class of hypotheses (hypothesis space) $\mathcal{H}$
- Choose a hypothesis $h \in \mathcal{H}$

# Example: what hypothesis class should we choose?



# Steps to solving a supervised learning problem

- Collect the data set (input-output pairs)
- Choose a class of hypotheses (hypothesis space) $\mathcal{H}$
- Choose a hypothesis $h \in \mathcal{H}$

# Linear model

- In linear regression, we consider the model $h_w$ has the form:

$$h_w(x) = w^\top x + b$$

  where $w = [w_1, \ldots, w_n]^\top \in \mathbb{R}^n$, $w_i$'s are called parameters or weights, and $b$ is called bias or intercept term.

- To simplify the notation, we can add a feature to the other $n$ features of $x$: $x \to [1; x] \in \mathbb{R}^{n+1}$, such that we will have

$$h_w(x) = w^\top x$$

  where $w$ and $x$ are vectors of size $n + 1$.

- How to choose $w$?

# Error (cost) Minimization

- Intuitively, $w$ should make the predictions of $h_w$ close to the true values $y$ on the data we have.
- Therefore, we will define an error function or cost function to measure the difference between our prediction and the true value
- We will choose $w$ such that the error function is minimized:

$$w = \arg\min_w \sum_{i=1}^{m} \ell(h_w(x_i), y_i), \tag{1}$$

  where $\ell$ is a loss function.

- (1) is the fundamental problem in machine learning!
  - How to choose $\ell$
  - How to solve (1)

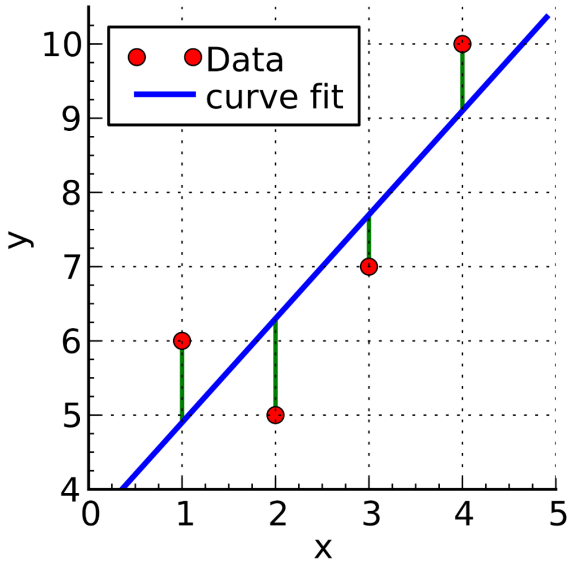
# Least squares

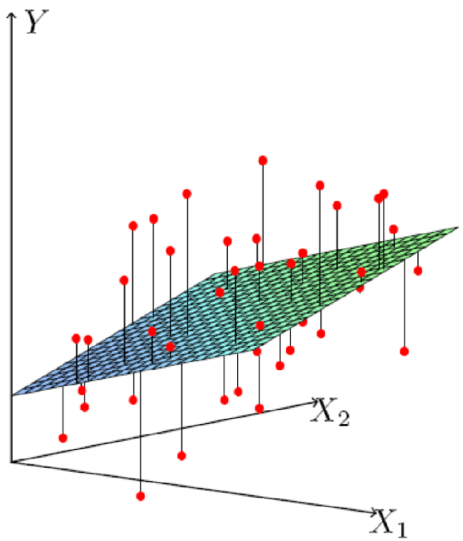
- We use the squared error to measure our prediction performance, and choose $w$ by minimizing the sum-of-squared errors:

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (h_w(x_i) - y_i)^2$$

(the $\frac{1}{2}$ is just for convenience)

- Fit by solving $\min_w J(w)$



# Solve $\nabla J(w) = 0$ using the definition

- In order to minimize $J(w)$, we need to solve $\nabla J(w) = 0$
- $\nabla J(w) = [\frac{\partial J(w)}{w_1}, \ldots, \frac{\partial J(w)}{w_n}]^\top$

$$\begin{aligned}
\frac{\partial J(w)}{w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (h_w(x_i) - y_i)^2 \\
&= \frac{1}{2} \cdot 2 \sum_{i=1}^{m} (h_w(x_i) - y_i) \frac{\partial}{\partial w_j} (h_w(x_i) - y_i) \\
&= \sum_{i=1}^{m} (h_w(x_i) - y_i) \frac{\partial}{\partial w_j} (w^\top x_i - y_i) \\
&= \sum_{i=1}^{m} (h_w(x_i) - y_i) x_{i,j}
\end{aligned}$$

  where $x_{i,j}$ is the $j$-th feature of the $i$-th instance.

- Setting all these partial derivatives to 0, we get a linear system with (n+1) equations and (n + 1) unknown variables.

# Solve $\nabla J(w) = 0$ using vector calculus

$$
\begin{aligned}
\nabla J(w) &= \nabla_w \left( \frac{1}{2} \sum_{i=1}^{m} (h_w(x_i) - y_i)^2 \right) \\
&= \nabla_w \left( \frac{1}{2} (Xw - y)^\top (Xw - y) \right) \\
&= \nabla_w \frac{1}{2} \left( w^\top X^\top X w - y^\top X w - w^\top X^\top y + y^\top y \right) \\
&= X^\top X w - X^\top y
\end{aligned}
$$

- Setting $\nabla J(w) = 0$ gives

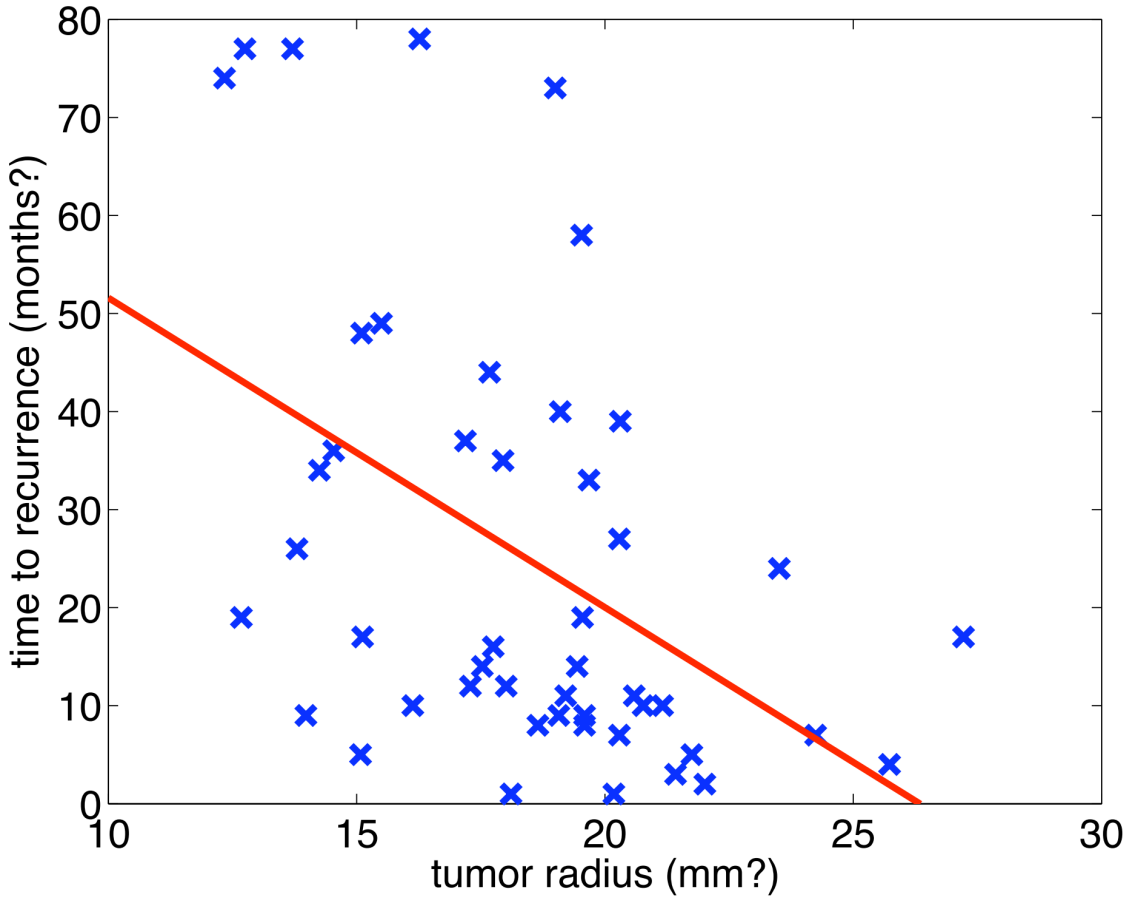
$$
\begin{aligned}
X^\top X w - X^\top y &= 0 \\
\Rightarrow X^\top X w &= X^\top y \\
\Rightarrow w &= (X^\top X)^{-1} X^\top y
\end{aligned}
$$

- The inverse exists if the columns of $X$ are linearly independent.

# Linear regression summary

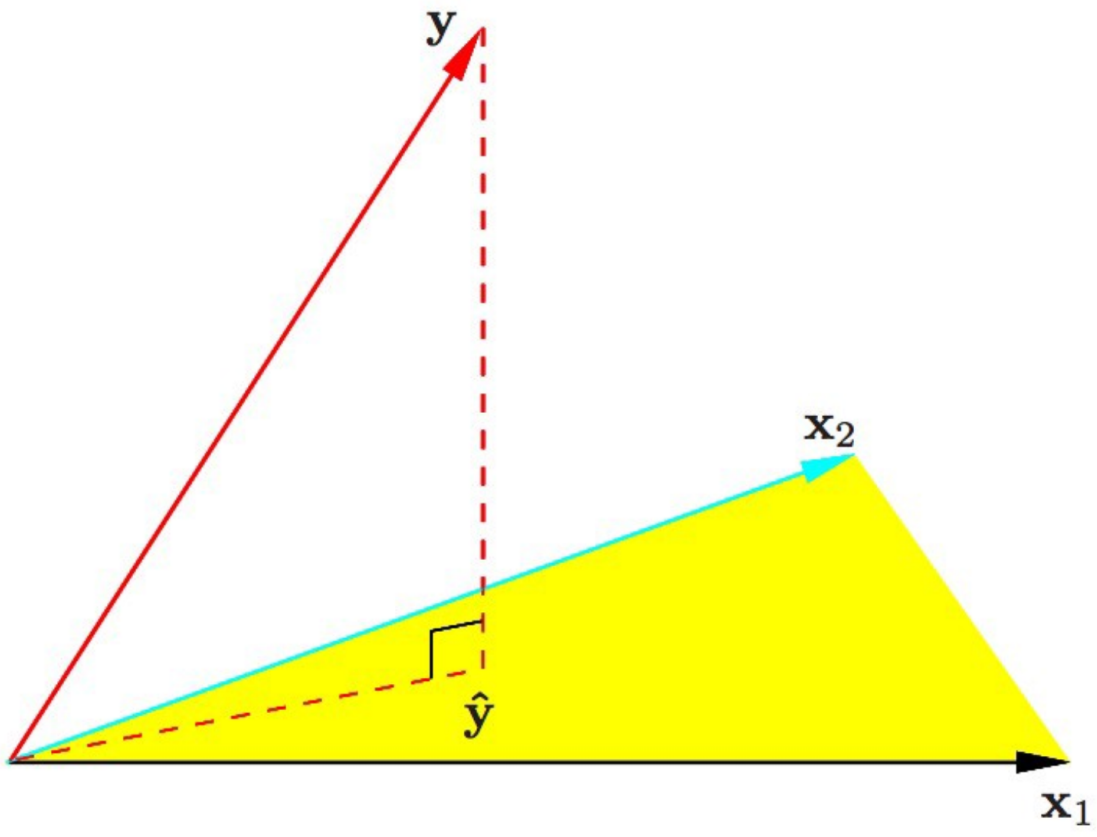
- The solution is $w = (X^\top X)^{-1} X^\top y$, where $X$ is the data matrix augmented with a column of ones, and $y$ is the column vector of target outputs.
- The number of data points should be at least larger or equal to the number of features to ensure a unique solution exists.
- An analytical, exact solution exists for linear regression, which is a rare case in machine learning.
- Demo!

# Predicting recurrence time based on tumor size




Slide credit: Doina Precup

# Geometric interpretation orthogonal projection




https://towardsdatascience.com/3-things-you-didnt-know-about-simple-linear-regression-b5a86362dd53

# Probabilistic perspective of linear regression

- Assume $y_i$ is a noisy target value, generated from a hypothesis $h_{\mathbf{w}}(\mathbf{x})$
- More specifically, assume that there exists $\mathbf{w}$ such that:

$$y_i = h_{\mathbf{w}}(\mathbf{x}_i) + \epsilon_i$$

  where $\epsilon_i$ is random variable (noise) drawn independently for each $\mathbf{x}_i$ according to some Gaussian (normal) distribution with mean zero and variance $\sigma$.
- How should we choose the parameter vector $\mathbf{w}$?

Slide credit: Doina Precup

# Bayes rule in machine learning

Let $h$ be a hypothesis and $D$ be the set of training data.
Using Bayes theorem, we have:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)},$$

where:

- $P(h)$ is the *prior probability of hypothesis $h$*
- $P(D) = \int_h P(D|h)P(h)$ is the probability of training data $D$ (normalization, independent of $h$)
- $P(h|D)$ is the probability of $h$ given $D$
- $P(D|h)$ is the probability of $D$ given $h$ (*likelihood of the data*)

Slide credit: Doina Precup

# Select a hypothesis by maximum a posteriori (MAP)

- What is the most probable hypothesis given the training data?
- *Maximum a posteriori (MAP)* hypothesis $h_{MAP}$:

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in \mathcal{H}} P(h|D) \\ &= \arg\max_{h \in \mathcal{H}} \frac{P(D|h)P(h)}{P(D)} \text{(using Bayes theorem)} \\ &= \arg\max_{h \in \mathcal{H}} P(D|h)P(h) \end{aligned}$$

  Last step is because $P(D)$ is independent of $h$ (so constant for the maximization)

- This is the Bayesian answer (more in a minute)

Slide credit: Doina Precup

# Maximum likelihood estimation

$$h_{MAP} = \arg\max_{h \in \mathcal{H}} P(D|h)P(h)$$

- If we assume $P(h_i) = P(h_j)$ (all hypotheses are equally likely a priori) then we can further simplify, and choose the *maximum likelihood (ML) hypothesis*:

$$h_{ML} = \arg\max_{h \in \mathcal{H}} P(D|h) = \arg\max_{h \in \mathcal{H}} L(h)$$

- Standard assumption: the training examples are *independently identically distributed (i.i.d.)*
- This alows us to simplify $P(D|h)$:

$$P(D|h) = \prod_{i=1}^{m} P(\langle \mathbf{x_i}, y_i \rangle | h) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i; h)P(\mathbf{x}_i)$$

Slide credit: Doina Precup

# The log trick

- We want to maximize:

$$L(h) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i; h)P(\mathbf{x}_i)$$

  This is a product, and products are hard to maximize!

- Instead, we will maximize $\log L(h)$! (the log-likelihood function)

$$\log L(h) = \sum_{i=1}^{m} \log P(y_i|\mathbf{x}_i; h) + \sum_{i=1}^{m} \log P(\mathbf{x}_i)$$

- The second sum depends on $D$, but not on $h$, so it can be ignored in the search for a good hypothesis

Slide credit: Doina Precup

# Maximum likelihood for regression

- Adopt the assumption that:

$$y_i = h_{\mathbf{w}}(\mathbf{x}_i) + \epsilon_i,$$

  where $\epsilon_i \sim \mathcal{N}(0, \sigma)$.
- The best hypothesis maximizes the likelihood of $y_i - h_{\mathbf{w}}(\mathbf{x}_i) = \epsilon_i$
- Hence,

$$L(\mathbf{w}) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_i - h_{\mathbf{w}}(\mathbf{x}_i)}{\sigma}\right)^2}$$

  because the noise variables $\epsilon_i$ are from a Gaussian distribution

Slide credit: Doina Precup

# Applying the log trick

$$\begin{aligned}\log L(\mathbf{w}) &= \sum_{i=1}^{m} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2}{\sigma^2}}\right) \\ &= \sum_{i=1}^{m} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{m} \frac{1}{2}\frac{(y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2}{\sigma^2}\end{aligned}$$

Maximizing the right hand side is the same as minimizing:

$$\sum_{i=1}^{m} \frac{1}{2}\frac{(y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2}{\sigma^2}$$

This is our old friend, the sum-squared-error function! (the constants that are independent of $h$ can again be ignored)

Slide credit: Doina Precup

# Maximum likelihood and linear regression

- Under the assumption that the training examples are i.i.d. and that we have Gaussian target noise, the maximum likelihood solution is equivalent to the least-squares solution for linear regression:

$$w = \arg\min_{w} \sum_{i=1}^{m} (h_w(x_i) - y_i)^2$$

- If the noise is not normally distributed, maximizing the likelihood will not be the same as minimizing the sum-squared error

- In practice, different loss functions are used depending on the noise assumption.

- Squared loss is the most popular one because it is the simplest one!

# Extending linear regression to more complex models

- Linear regression should be the first thing you try for real-valued outputs!

- Two possible solutions:

  1. Transform the data – apply a (nonlinear) transformation $x \rightarrow \phi(x)$, then do linear regression in the transformed space. (our focus in this lecture)
     - e.g., log, exp, sin, cos
     - polynomial transformation: cross-terms, higher-order terms...
     - basis expansions: Gaussian basis, polynomial basis...
     - Dummy coding of categorical features

  - Use a different hypothesis class (e.g. non-linear functions)

# Linear models in general

- Given a set of examples $\langle \mathbf{x}_i, y_i \rangle_{i=1...m}$, we fit a hypothesis

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{k=0}^{K-1} w_k \phi_k(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

  where $\phi_k$ are called basis functions
- The best $\mathbf{w}$ is considered the one which minimizes the sum-squared error over the training data:

$$\sum_{i=1}^{m} (y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2$$

- We can find the best $\mathbf{w}$ in closed form:

$$\mathbf{w} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{y}$$

  or by other methods (e.g. gradient descent - as will be seen later)

Slide credit: Doina Precup

# Linear models in general

- By linear models, we mean that the hypothesis function $h_{\mathbf{w}}(\mathbf{x})$ is a *linear function of the parameters* $\mathbf{w}$
- This *does not mean the* $h_{\mathbf{w}}(\mathbf{x})$ *is a linear function of the input vector* $\mathbf{x}$ (e.g., polynomial regression)
- In general

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{k=0}^{K-1} w_k \phi_k(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

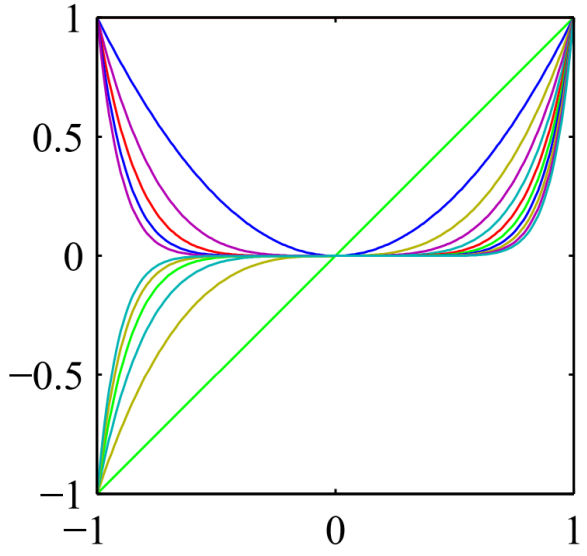
  where $\phi_k$ are called *basis functions*
- Usually, we will assume that $\phi_0(\mathbf{x}) = 1, \forall \mathbf{x}$, to create a bias term
- The hypothesis can alternatively be written as:

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{\Phi w}$$

  where $\mathbf{\Phi}$ is a matrix with one row per instance; row $j$ contains $\phi(\mathbf{x}_j)$.
- Basis functions are *fixed*

Slide credit: Doina Precup

# Example basis functions: Polynomial

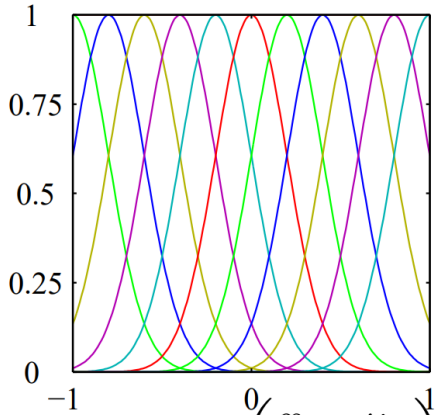



$$\phi_k(x) = x^k$$

"Global" functions: a small change in $x$ may cause large change in the output of many basis functions

Slide credit: Doina Precup

# Example basis functions: Gaussian

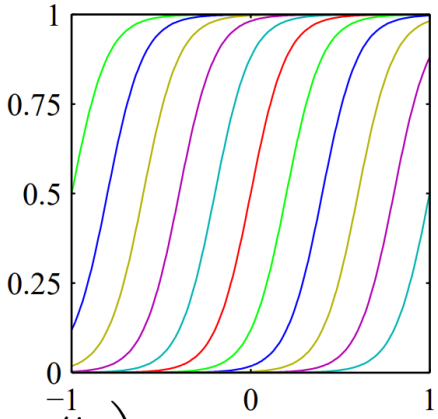



$$\phi_k(x) = \exp\left(\frac{x - \mu_k}{2\sigma^2}\right)$$

- $\mu_k$ controls the position along the x-axis
- $\sigma$ controls the width (activation radius)
- $\mu_k$, $\sigma$ fixed for now (later we discuss adjusting them)
- Usually thought as "local" functions: if $\sigma$ is relatively small, a small change in $x$ only causes a change in the output of a few basis functions (the ones with means close to $x$)

Slide credit: Doina Precup

# Example basis functions: Sigmoidal




$$\phi_k(x) = \sigma\left(\frac{x - \mu_k}{s}\right) \quad \text{where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$

- $\mu_k$ controls the position along the x-axis
- $s$ controls the slope
- $\mu_k$, $s$ fixed for now (later we discuss adjusting them)
- "Local" functions: a small change in $x$ only causes a change in the output of a few basis (most others will stay close to 0 or 1)

Slide credit: Doina Precup

# Polynomial regression

- Given data: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$.
- Suppose we want a degree-$d$ polynomial fit.
- Let $\mathbf{y}$ be as before and let

$$\mathbf{X} = \begin{bmatrix} x_1^d & \ldots & x_1^2 & x_1 & 1 \\ x_2^d & \ldots & x_2^2 & x_2 & 1 \\ \vdots & & \vdots & \vdots & \vdots \\ x_m^d & \ldots & x_m^2 & x_m & 1 \end{bmatrix}$$

- Solve the linear regression $\mathbf{X}\mathbf{w} \approx \mathbf{y}$.

Slide credit: Doina Precup

# Example: second-order polynomial regression

$$\mathbf{X} = \begin{bmatrix} 0.75 & 0.86 & 1 \\ 0.01 & 0.09 & 1 \\ 0.73 & -0.85 & 1 \\ 0.76 & 0.87 & 1 \\ 0.19 & -0.44 & 1 \\ 0.18 & -0.43 & 1 \\ 1.22 & -1.10 & 1 \\ 0.16 & 0.40 & 1 \\ 0.93 & -0.96 & 1 \\ 0.03 & 0.17 & 1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

Slide credit: Doina Precup

# Example: second-order polynomial regression

$$\mathbf{X}^T\mathbf{X} =$$

$$\begin{bmatrix} 0.75 & 0.01 & 0.73 & 0.76 & 0.19 & 0.18 & 1.22 & 0.16 & 0.93 & 0.03 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.75 & 0.86 & 1 \\ 0.01 & 0.09 & 1 \\ 0.73 & -0.85 & 1 \\ 0.76 & 0.87 & 1 \\ 0.19 & -0.44 & 1 \\ 0.18 & -0.43 & 1 \\ 1.22 & -1.10 & 1 \\ 0.16 & 0.40 & 1 \\ 0.93 & -0.96 & 1 \\ 0.03 & 0.17 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4.11 & -1.64 & 4.95 \\ -1.64 & 4.95 & -1.39 \\ 4.95 & -1.39 & 10 \end{bmatrix}$$

Slide credit: Doina Precup

# Example: second-order polynomial regression

$$\mathbf{X}^T\mathbf{y} =$$

$$\begin{bmatrix} 0.75 & 0.01 & 0.73 & 0.76 & 0.19 & 0.18 & 1.22 & 0.16 & 0.93 & 0.03 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$= \begin{bmatrix} 3.60 \\ 6.49 \\ 8.34 \end{bmatrix}$$

Slide credit: Doina Precup

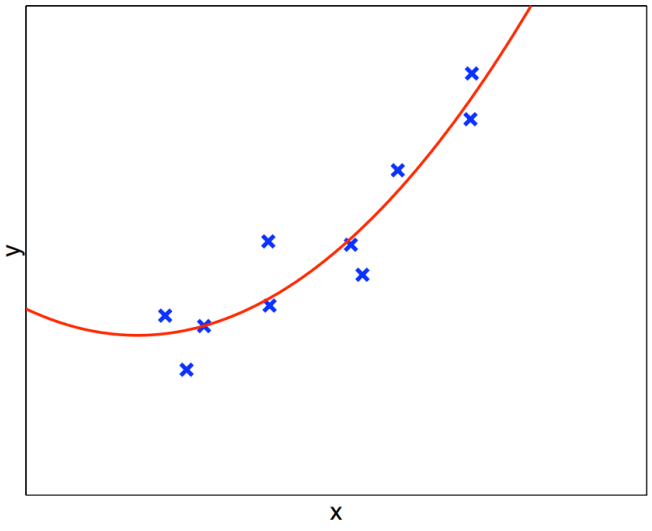
# Example: second-order polynomial regression

$$\begin{aligned}\mathbf{w} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ &= \begin{bmatrix} 4.11 & -1.64 & 4.95 \\ -1.64 & 4.95 & -1.39 \\ 4.95 & -1.39 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 3.60 \\ 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 0.68 \\ 1.74 \\ 0.73 \end{bmatrix}\end{aligned}$$

So the best order-2 polynomial is $y = 0.68x^2 + 1.74x + 0.73$.

Slide credit: Doina Precup

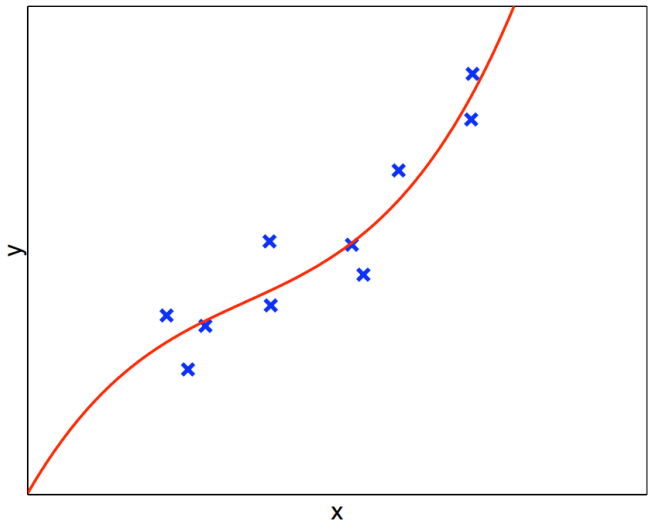
# Higher-order polynomial regression: Order-2 fit




Is this a better fit to the data?

Slide credit: Doina Precup

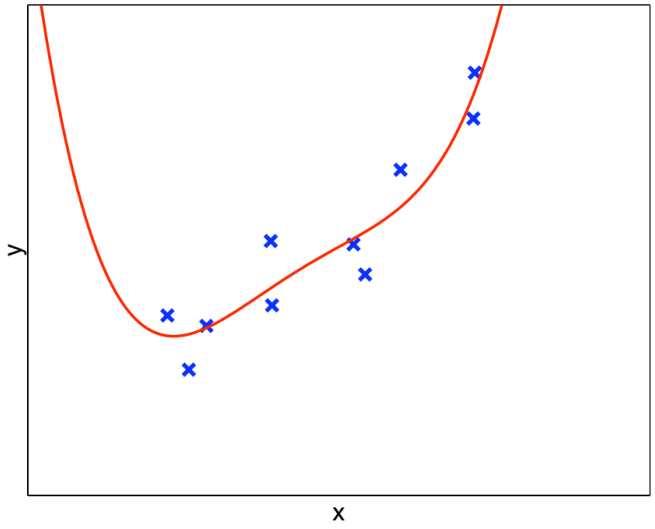
# Higher-order polynomial regression: Order-3 fit




Is this a better fit to the data?

Slide credit: Doina Precup

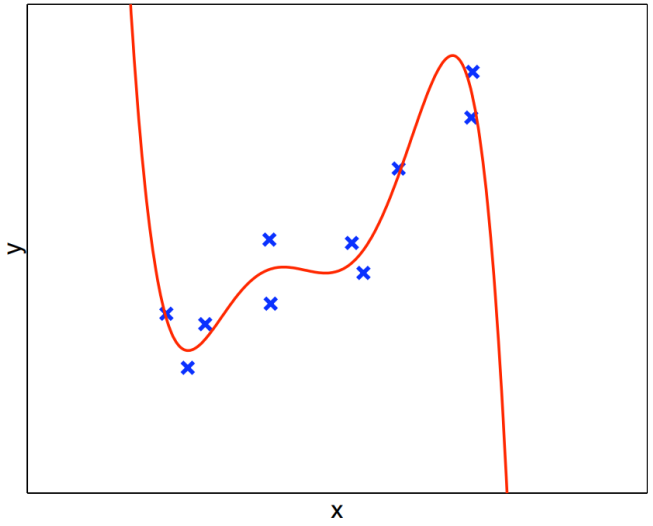
# Higher-order polynomial regression: Order-4 fit




Is this a better fit to the data?

Slide credit: Doina Precup

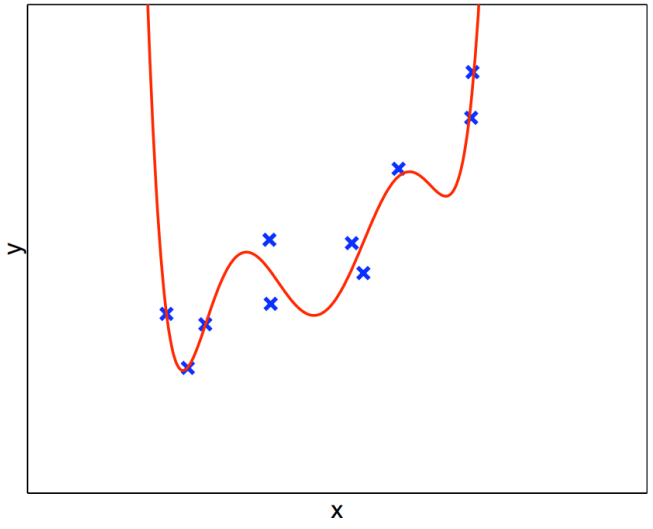
# Higher-order polynomial regression: Order-5 fit




Is this a better fit to the data?

Slide credit: Doina Precup

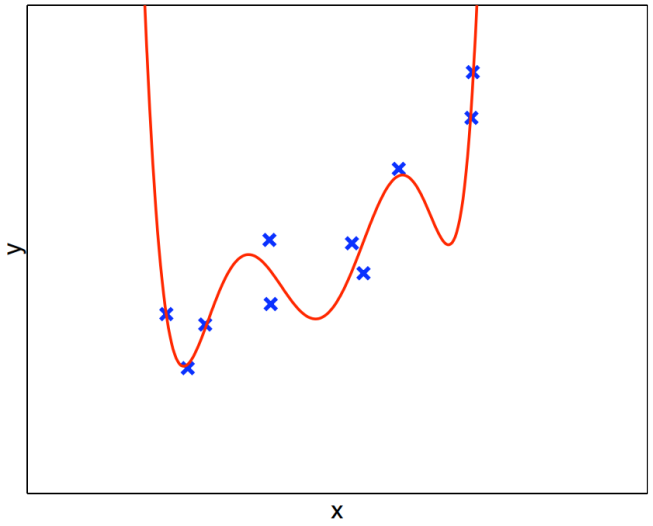
# Higher-order polynomial regression: Order-6 fit




Is this a better fit to the data?

Slide credit: Doina Precup

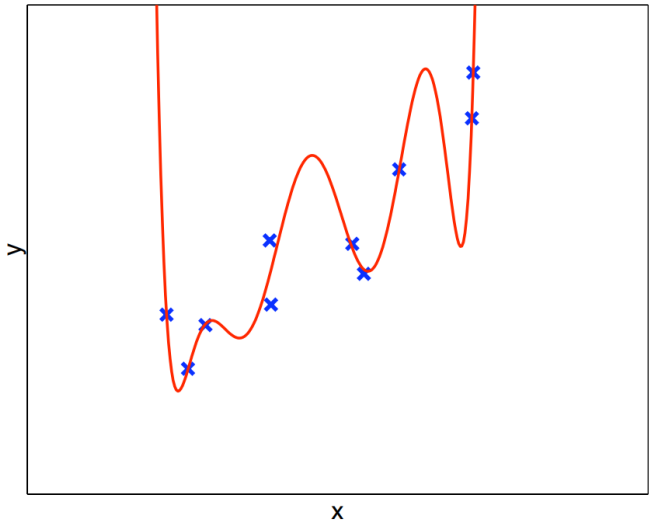
# Higher-order polynomial regression: Order-7 fit




Is this a better fit to the data?

Slide credit: Doina Precup

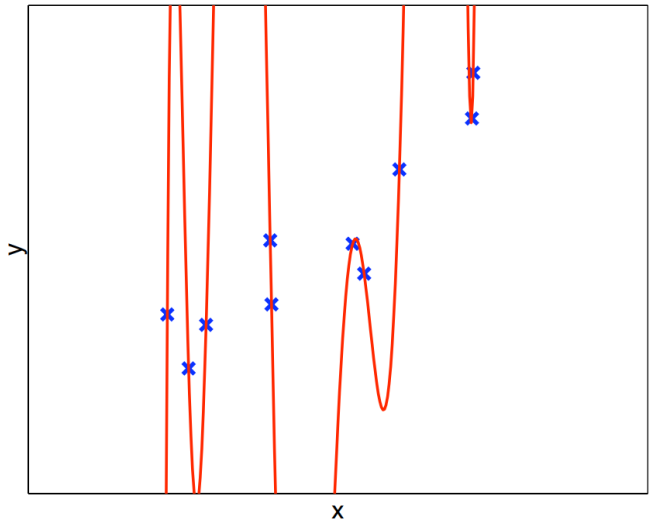
# Higher-order polynomial regression: Order-8 fit




Is this a better fit to the data?

Slide credit: Doina Precup

# Higher-order polynomial regression: Order-9 fit




Is this a better fit to the data?

Slide credit: Doina Precup

# Overfitting and model selection

- As long as the the hypothesis is complicated enough, we can always achieve zero training loss (e.g., memorize all the data)
- Minimizing the training loss does NOT indicate a good test performance
- Overfitting: A (complicated) model fits data too well, but cannot generalize at all to new data
- An extremely important problem for (almost all) machine learning algorithms