Regression - predicting continuous outputs.
Examples:
- Future stock prices.
- Tracking - object location in the next time-step.
- Housing prices.
- Crime rates.

We don’t just have infinite number of possible answers, we assume a simple geometry - closer is better.

We will focus on *linear* regression models.
What do I need in order to make predictions? **In linear regression**

- **Inputs (features)** $x$ ($x$ for vectors). A vector $x \in \mathbb{R}^d$
- **Output (dependent variable)** $y$. $y \in \mathbb{R}$
- **Training data.** $(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})$
- **A model/hypothesis class**, a family of functions that represents the relationship between $x$ and $y$.
  
  $$f_w(x) = w_0 + w_1 x_1 + ... w_d x_d \text{ for } w \in \mathbb{R}^{d+1}$$
- **A loss function** $\ell(y, \hat{y})$ that assigns a cost to each prediction.
  
  $$L_2(y, \hat{y}) = (y - \hat{y})^2, \quad L_1(y, \hat{y}) = |y - \hat{y}|$$
- **Optimization** - a way to minimize the loss objective.
  
  Analytic solution, convex optimization
Linear model seems very limited, for example

is not close to linear.

In linear model we mean linear in parameters not the inputs!

\(^1\)Images from Bishop
Any (fixed) transformation $\phi(x) \in \mathbb{R}^d$ we can run linear regression with features $\phi(x)$.

Example: Polynomials $w_0 + w_1 x + \ldots + w_d x^d$ are a linear (in $w$) model.

Feature engineering - design good features and feed them to a linear model.

Commonly replaced with deep models that learn the features as well.
Most common loss is $L_2(y, \hat{y}) = (y - \hat{y})^2$.

Easy to optimize (convex, analytic solution), well understood, harshly punishes large mistakes. Can be good (e.g. financial predictions) or bad (outliers).

The optimal prediction w.r.t $L_2$ loss is the conditional mean $E[y|x]$ (show!).

Equivalent to assuming Gaussian noise (more on that later).
Another common loss is $L_1(y, \hat{y}) = |y - \hat{y}|$.

Easyish to optimize (convex), well understood, Robust to outliers.

The optimal prediction w.r.t $L_2$ loss is the conditional median (show!).

Equivalent to assuming Laplace noise.

You can combine both - Huber loss.
Deriving and analyzing the optimal solution:

Notation: We can include the bias into $\mathbf{x}$ by adding 1, $\mathbf{x}^{(i)} = [1, x_1^{(i)}, \ldots, x_d^{(i)}]$. Prediction is $\mathbf{x}^T \mathbf{w}$.

Target vector $\mathbf{y} = [y^{(1)}, \ldots, y^{(N)}]^T$.

Feature vectors $\mathbf{f}^{(j)} = [\mathbf{x}_j^{(1)}, \ldots, \mathbf{x}_j^{(N)}]^T$.

Design matrix $\mathbf{X}$, $\mathbf{X}_{ij} = \mathbf{x}_j^{(i)}$.

Rows correspond to data points, columns to features.
Theorem

The optimal $w$ w.r.t $L_2$ loss, $w^* = \arg \min \sum_{i=1}^{N} (y^{(i)} - w^T x^{(i)})^2$ is $w^* = (X^T X)^{-1} X^T y$.

Proof (sketch): Our predictions vector are $\hat{y} = Xw$ and the total loss is $L(w) = ||y - \hat{y}||^2 = ||y - Xw||^2$.

Rewriting $L(w) = ||y - Xw||^2 = (y - Xw)^T (y - Xw) = y^T y + w^T X^T X w - 2w^T X^T y$.

$\nabla L(w^*) = 2X^T X w^* - 2Xy = 0 \Rightarrow X^T X w^* = X^T y$.

If the features aren’t linearly dependent $X^T X$ is invertible.

Never actually invert! Use linear solvers (Conjugate gradients, Cholesky decomp,...)
Some intuition: Our predictions are $\hat{y} = Xw^*$ and we have $X^TXw^* = X^Ty$.

Residual $r = y - \hat{y} = y - Xw^*$, so $X^Tr = 0$.

This means $r$ is orthogonal to $f^{(1)}, ..., f^{(d)}$ (and zero mean).

Geometrically we are projecting $y$ to the subspace spun by the features.
Assume the features have zero mean $\sum_j f_j^{(i)} = 0$, in this case $[X^T X]_{ij} = \text{cov}(f^{(i)}, f^{(j)})$ and $[X^T y]_j = \text{cov}(f^{(j)}, y)$.

If the covariance is diagonal (data-whitening, see tutorial), $\text{var}(f^{(j)}) \cdot w_j = \text{cov}(f^{(j)}, y) \Rightarrow w_j = \frac{\text{cov}(f^{(j)}, y)}{\text{var}(f^{(j)})}$.

Good feature $=$ large signal to noise ratio (loosely speaking).
Back to our simple example - let's fit a polynomial of degree $M$. 
Back to our simple example - let's fit a polynomial of degree $M$. 

![Graphs showing polynomial fitting for different degrees $M$](image)
Generalization = models ability to predict the held out data.

- Model with $M = 1$ underfits (cannot model data well).
- Model with $M = 9$ overfits (it models also noise).
- Not a problem if we have lots of training examples (rule-of-thumb $10 \times \text{dim}$)
- Simple solution - model selection (validation/cross-validation)
Observation: Overfitting models term to have large norm.

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 6$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>-1.27</td>
<td></td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$w_3^*$</td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td>-231639.30</td>
<td>640042.26</td>
<td></td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td>-1061800.52</td>
<td></td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td>1042400.18</td>
<td></td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td>-557682.99</td>
<td></td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td>125201.43</td>
<td></td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution: Regularizer $R(w)$ penalizing large norm, $w^* = \arg\min_w L_S(w) + R(w)$.

Commonly use $R(w) = \frac{\lambda}{2} \|w\|_2^2 = \frac{\lambda}{2} w^T w = \frac{\lambda}{2} \sum w_j^2$
$L_2$ regularization $R(w) = \frac{\lambda}{2} w^T w$

Objective $\sum_i (w^T x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2} w^T w$.

Analytic solution $w^* = (X^T X + \lambda I)^{-1} X y$ (show!)

Can show equivalence to Gaussian prior.

Normaly we do not regularize the bias $w_0$.

Use validation/cross-validation to find a good $\lambda$. 
Regularization

The model

Optimization

Generalization

Probabilistic viewpoint

Regularization

\[ \ln \lambda = -18 \]

\[ \ln \lambda = 0 \]

\[ E_{RMS} \]

Training
Test

CSC411-Lec2
Another common regularizer: $L_1$ regularization

$$R(w) = \lambda ||w||_1 = \lambda \sum |w_i|$$

Convex (SGD) but no analytic solution

Tends to induce *sparse* solutions.

Can show equivalence to Laplacian prior.
Probabilistic viewpoint: Assume \( p(y^{(i)}|x^{(i)}) = w^T x^{(i)} + \epsilon_i \) and \( \epsilon_i \) are i.i.d \( \epsilon_i \sim \mathcal{N}(0, \sigma^2) \). \( p(y|x) = \mathcal{N}(w^T x, \sigma^2) = \frac{\exp\left(-\frac{||y-w^T x||^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}} \).

\( w \) parametrizes a distribution. Which distribution to pick? Maximize the likelihood of the observation.

Log-likelihood
\[
\log(p(y^{(1)}, ..., y^{(N)}|x^{(1)}, ..., x^{(N)}; w)) = \log \left( \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}; w) \right) = \sum_{i=1}^{N} \log \left( p(y^{(i)}|x^{(i)}; w) \right).
\]

Linear Gaussian model
\[
\Rightarrow \log(p(y|x; w)) = -\frac{||y-w^T x||^2}{2\sigma^2} - 0.5 \log(2\pi\sigma^2)
\]

maximum likelihood = minimum \( L_2 \) loss.
”When you hear hoof-beats, think of horses not zebras” Dr. Theodore Woodward.

ML finds a model that makes the observation likely $P(data|w)$, we want the most probable model $p(w|data)$.

Bayes formula $P(w|y, X) = \frac{P(y|w, X)p(w)}{p(y|X)} \propto P(y|w, X)p(w)$

Need prior $p(w)$ - what model is more likely?

MAP=Maximum a posteriori estimator

$w_{MAP} = \arg \max P(w|y, X) = \arg \max P(y|w, X)p(w)$

$= \arg \max \log(P(y|w, X)) + \log(p(w))$
Convenient prior (conjugate): \( p(w) = \mathcal{N}(0, \sigma_w^2) \)

\[
\mathbf{w}_{\text{map}} = \arg \max \log(P(y|w, X)) + \log(p(w)) = -\frac{||y - \mathbf{w}^T \mathbf{x}||^2}{2\sigma^2} - \frac{||\mathbf{w}||^2}{2\sigma_w^2}
\]

\( L_2 \) regularization = Gaussian prior.
Recap:

- Linear models benefit: Simple, fast (test time), generalize well (with regularization).
- Linear models limitations: Performance crucially depends on good features.
- Modeling questions - loss and regularizer (and features)
- $L_2$ loss and regularization - analytical solution, otherwise stochastic optimization (next week).
- Difficulty with multimodel distribution - discretization might work much better.