Lecture Recording

- **Note: These lectures will be recorded and posted onto the IMPRS website**

- Dear participants,

- We will record all lectures on *"Making sense of data: introduction to statistics for gravitational wave astronomy"*, including possible Q&A after the presentation, and we will make the recordings publicly available on the IMPRS lecture website at:

  - https://imprs-gw-lectures.aei.mpg.de/2021-making-sense-of-data/

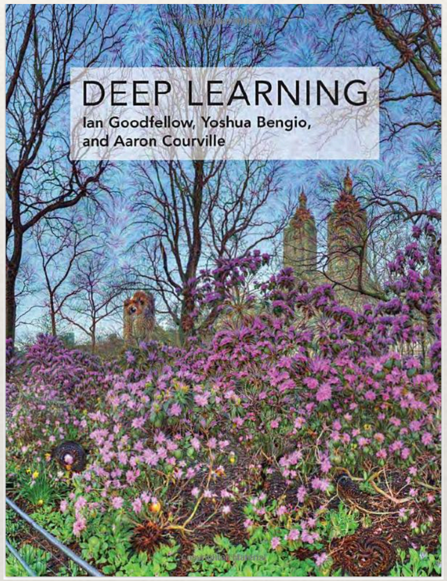
- By participating in this Zoom meeting, you are giving your explicit consent to the recording of the lecture and the publication of the recording on the course website.

Outline of course

- **Part 4 (week 4): Introduction to machine learning**
  - Lecture 10: Machine learning
  - Lecture 11: Neural networks and deep learning
  - Lecture 12: Machine learning for GW astronomy
  - Practical: GW search and parameter estimation using machine learning

References

- **Textbook: "Deep Learning" by Goodfellow, Bengio, and Courville**
  - Free online at https://www.deeplearningbook.org
  - Course covers parts of Chapters 5, 6, 9, 20



- **PyTorch**
  - machine learning framework for practical part
  - many tutorials at https://pytorch.org

Making sense of data: introduction to statistics for gravitational-wave astronomy

Lecture 10: Machine Learning

AEI IMPRS Lecture Course

Stephen Green stephen.green@aei.mpg.de

Introduction to Machine Learning

- **Machine learning uses computers to learn patterns from data.**
  - Typically used to solve problems that are hard to program in conventional ways. Instead, train by example.
- Examples:
  - **Classification:** Learn a function mapping data into a particular category

$$f : \mathbb{R}^n \to \{1, \ldots, k\}$$

E.g., recognize handwritten digits

8, 8, 1, 5, 1
4, 4, 7, 4, 9

Machine Learning Tasks

- Examples:
  - **Regression:** Learn a function predicting real-valued quantities

    $$f : \mathbb{R}^n \to \mathbb{R}^m$$

    E.g., What are the physical parameters characterizing a binary merger?

  - **Sampling:** Generate new samples similar to training examples.
  - **Denoising:** Given noisy data $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$, predict clean data $\boldsymbol{x} \in \mathbb{R}^n$: $\quad p(\boldsymbol{x} \,|\, \tilde{\boldsymbol{x}})$
  - **Density estimation:** Given training examples $\boldsymbol{x} \in \mathbb{R}^n$ learn a probability density function $p(\boldsymbol{x})$.
  - **Game playing:** Given a game configuration, what is the best move to make?

Performance Measures

- **For each task, it is necessary to specify some quantitative measure of performance:**
    - for **classification**, the accuracy (the fraction of examples that produce the correct output)
    - for **density estimation**, the log probability assigned to examples
    - for **regression**, the mean squared error
- We are usually interested in how the machine learning algorithm performs on data that have not been seen before: Evaluate performance on a **test set** that is different from the **training set**.

Types of Learning Algorithms

- Typically we have a dataset $\{\boldsymbol{x}^{(i)}\}$ consisting of many data points $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$. The data points may or may not have associated labels $\boldsymbol{y}^{(i)} \in \mathbb{R}^m$.
  - **Unsupervised:** learn $p(\boldsymbol{x})$
    - Examples: density estimation, sampling
  - **Supervised:** learn $p(\boldsymbol{y} \mid \boldsymbol{x})$
    - Examples: regression, classification

somewhat hazy distinction, e.g., learning $p(\boldsymbol{y}, \boldsymbol{x})$

- **Reinforcement** learning allows the algorithm to interact with the environment and produce new samples (e.g., game playing).

Maximum likelihood estimation

- Consider a set of N independent examples $\boldsymbol{x}^{(i)} \sim p_{\text{data}}(\boldsymbol{x})$ drawn from the data-generating distribution.
- **Unsupervised learning:** Let $p_{\text{model}}(\boldsymbol{x};\boldsymbol{\theta})$ be a parametric family of model probability distributions. Choose $\boldsymbol{\theta}$ such that this becomes a good approximation to $p_{\text{data}}(\boldsymbol{x})$.
- **Maximum likelihood estimator** is

$$
\begin{aligned}
\boldsymbol{\theta}_{\mathrm{ML}} &= \arg\max_{\theta} p_{\text{model}}(\boldsymbol{X};\boldsymbol{\theta}) \\
&= \arg\max_{\theta} \prod_{i=1}^{N} p_{\text{model}}(\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \\
&= \arg\max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \\
&= \arg\max_{\theta} \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \log p_{\text{model}}(\boldsymbol{x};\boldsymbol{\theta})
\end{aligned}
$$

- Equivalent to minimizing **KL divergence** or **cross-entropy** between p_{data} and p_{model}.

Conditional Estimation

- **Supervised learning:** Estimate a conditional probability $p_{\text{model}}(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$
- Generalize the maximum likelihood estimator:

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(\boldsymbol{y}^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\theta} \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}, \boldsymbol{y})} \log p_{\text{model}}(\boldsymbol{y} | \boldsymbol{x}; \boldsymbol{\theta})$$

- This is one of the most common situations.

Example: Linear regression

- Suppose we have labelled data $(\boldsymbol{x}^{(i)}, y^{(i)})$.
- Let $p(y\,|\,\boldsymbol{x}) = \mathcal{N}\left(\mu(\boldsymbol{x}), \sigma^2\right)(y)$ where $\mu(\boldsymbol{x}) = \boldsymbol{\theta} \cdot \boldsymbol{x}$; σ fixed.
- Using the PDF $p(y|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu(\boldsymbol{x}))^2}{2\sigma^2}\right)$

  we obtain the loss function

$$J(\theta) = -\sum_{i=1}^{N} \log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

$$= \frac{N}{2}\log 2\pi\sigma^2 + \boxed{\sum_{i=1}^{N} \frac{(y^{(i)} - \mu(\boldsymbol{x}^{(i)}))^2}{2\sigma^2}}$$

$\propto$ mean squared error

- Can solve exactly $\nabla_{\boldsymbol{\theta}} J = 0 \implies \boldsymbol{\theta}_{\mathrm{ML}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$

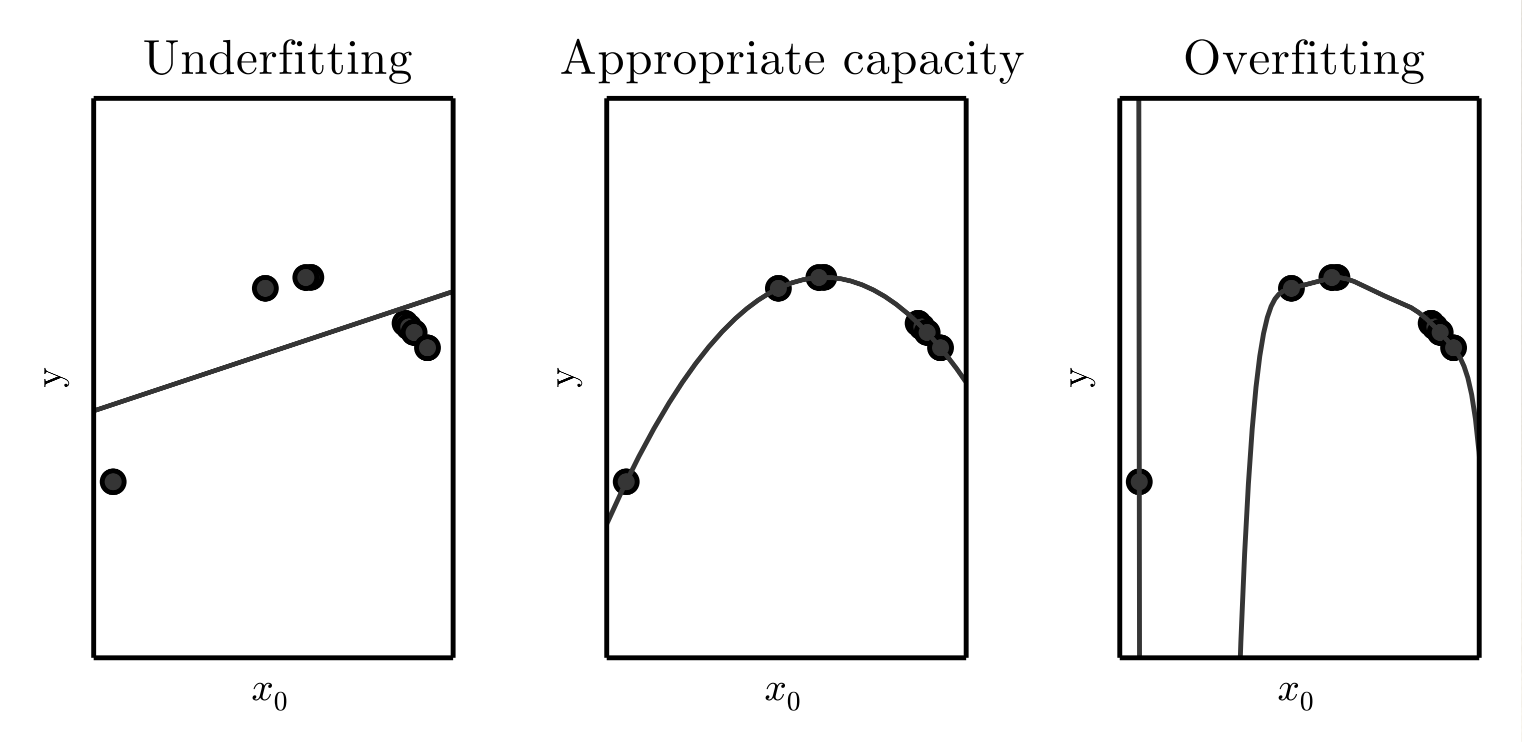
More general regression

- More generally $\mu(\boldsymbol{x})$ does not have to be linear. We can increase the **representational capacity** of the model by using more complicated functions.
    - E.g., polynomial $$\mu(x) = b + \sum_{i=1}^{k} w_i x^i$$ "hyperparameter" (can still solve in closed form)
    - E.g. nonparametric regression

      nearest neighbor: For any $\boldsymbol{x}$, find the nearest $\boldsymbol{x}^{(i)}$ in the training set and return $y^{(i)}$.
    - E.g., neural network (next lecture)
- Not all models can be optimized in closed form. The optimization algorithm may be imperfect, so the **effective capacity** is lower than the representational capacity.

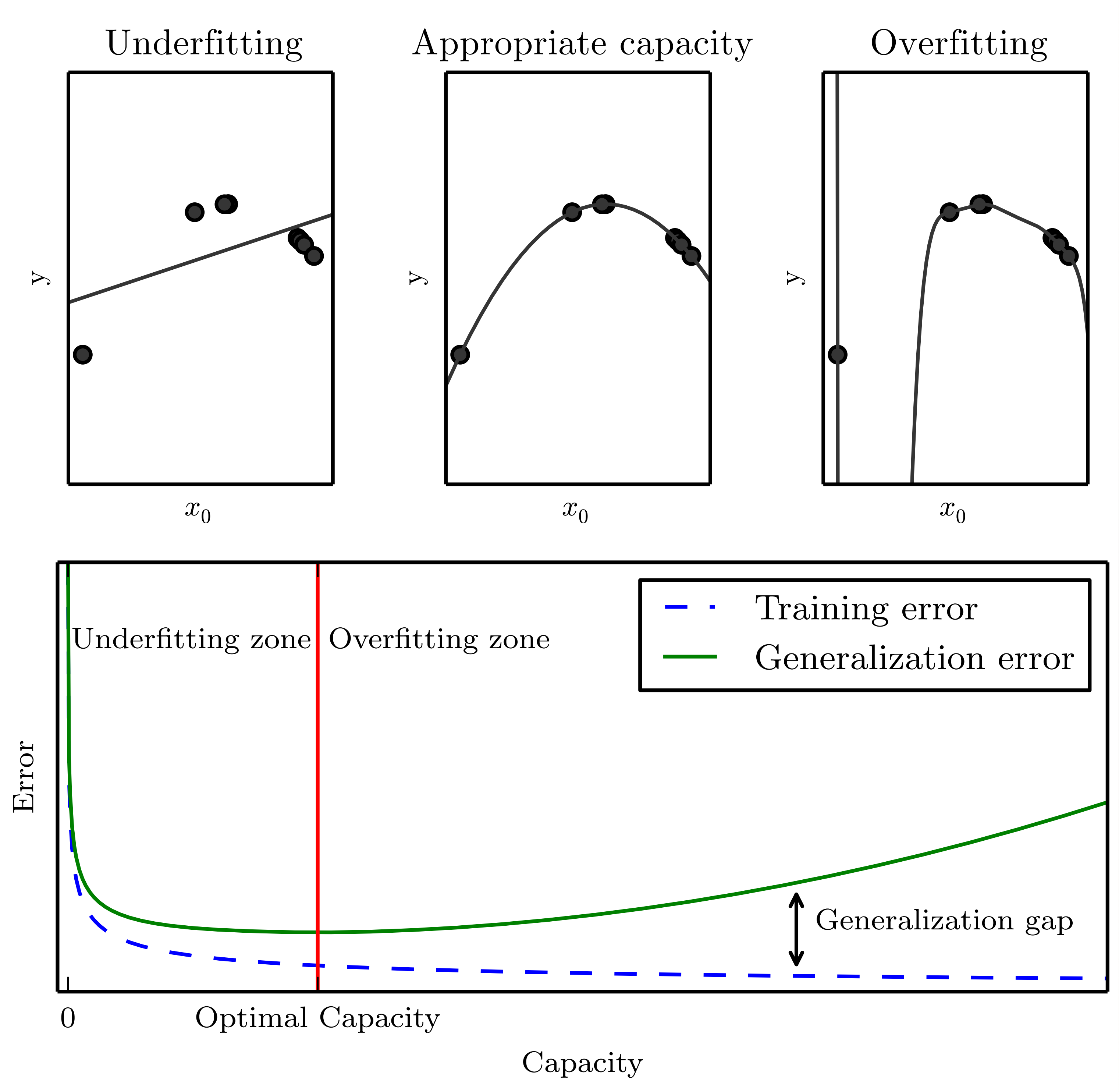
Overfitting and underfitting

- Higher capacity models run the risk of **overfitting.** The algorithm must perform well not just on data used for training, but also on new, previously unseen inputs (test data). This is called **generalization.**
- Training and test examples should be **independent and identically distributed (i.i.d.)**, i.e., drawn from the same data-generating distribution p_{data}



Goodfellow et al (2016)

Overfitting and underfitting



- Capacity should be chosen to minimize generalization error.
- Depends also on the size of the training set.

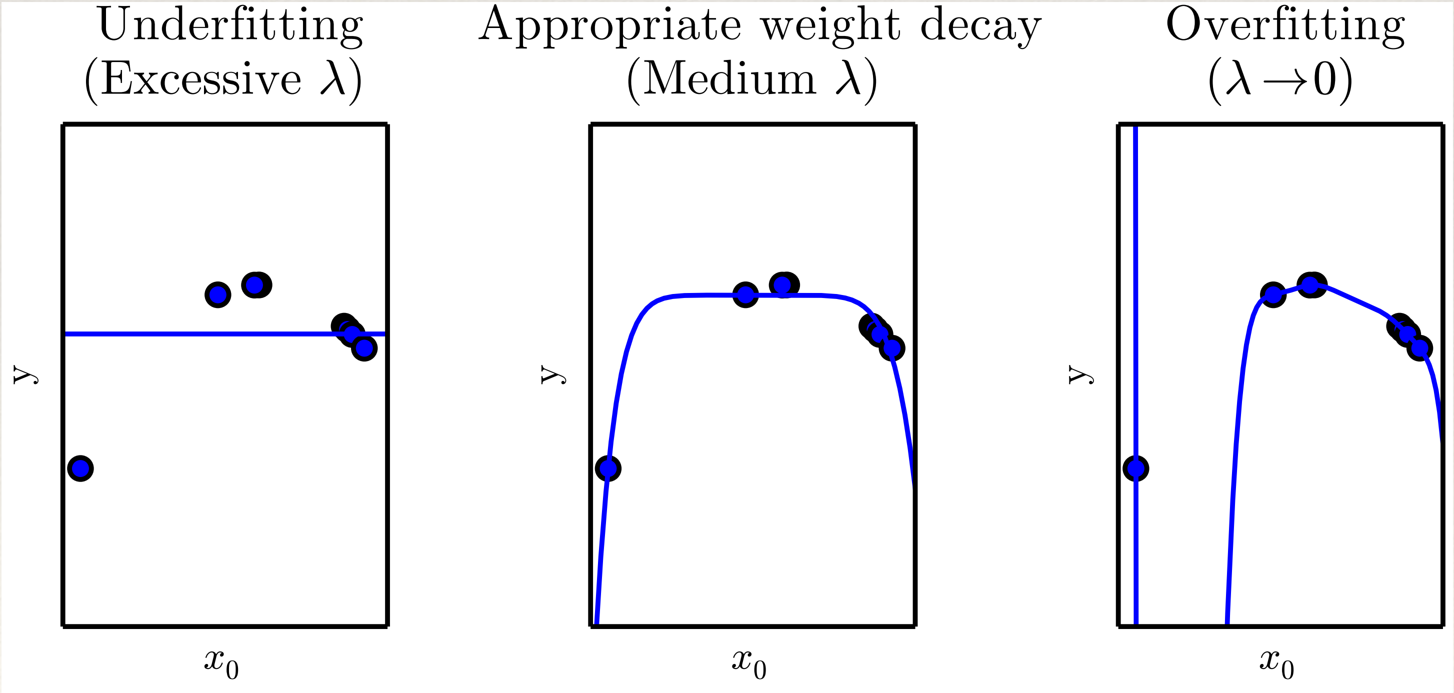
Goodfellow et al (2016)

Regularization

- One way to improve generalization is to build in preferences for certain values of the parameters $\boldsymbol{\theta}$, without changing the representational capacity.

- Add a **regularizer** to the loss function.

Weight decay: $J(\boldsymbol{\theta}) = \mathrm{MSE} + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta}$ preference for small values of $\boldsymbol{\theta}$



Goodfellow et al (2016)

But do we still have a probabilistic interpretation of this loss?

Bayesian statistics for model parameters

- The maximum likelihood objective picks out a single choice of parameters $\boldsymbol{\theta}_{\mathrm{ML}}$ corresponding to the maximum of $p(\boldsymbol{X}\,|\,\boldsymbol{\theta})$.
- We can also treat $\boldsymbol{\theta}$ in a Bayesian way:
  - Specify a prior $p(\boldsymbol{\theta})$
  - Obtain the posterior using Bayes' rule $p(\boldsymbol{\theta}|\boldsymbol{X}) = \dfrac{p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{X})}$
- This incorporates the **uncertainty** associated to the choice of $\boldsymbol{\theta}$.
- The **prior** acts as a regularizer.

Example: Bayesian linear regression

- As before we take a Gaussian likelihood $p(\boldsymbol{y}\,|\,\boldsymbol{X},\boldsymbol{w}) = \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, I)(\boldsymbol{y})$
- Also take a Gaussian prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0)$
- Exercise: show that the posterior is also Gaussian, of the form

$$p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu}_m)^\top \Lambda_m^{-1}(\boldsymbol{w}-\boldsymbol{\mu}_m)\right)$$

$$\boldsymbol{\Lambda}_m = (\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{\Lambda}_0^{-1})^{-1} \qquad \boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m(\boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{\Lambda}_0^{-1}\boldsymbol{\mu}_0)$$

Maximum a posteriori estimation

- To obtain a point estimate that still takes into account prior, we can take the maximum of the posterior distribution over $\boldsymbol{\theta}$,

$$\begin{aligned}\boldsymbol{\theta}_{\mathrm{MAP}} &= \arg\max_{\theta} p(\boldsymbol{\theta}|\boldsymbol{x}) \\ &= \arg\max_{\theta} \left(\log p(\boldsymbol{x}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right)\end{aligned}$$

For $p(\boldsymbol{w}) = \mathcal{N}(0, \boldsymbol{I}/\lambda)$ this term $\longrightarrow \lambda \boldsymbol{w}^\top \boldsymbol{w}$

- MAP Bayesian inference with a Gaussian weight prior corresponds to weight decay. More generally, MAP provides a way to interpret regularization terms.

Example: Logistic regression

- If instead of estimating real-valued quantity *y*, we are interested in a binary classification problem with $y \in \{0,1\}$, we can use **logistic regression**.
- Use a logistic sigmoid function $\sigma(u) = \dfrac{1}{1 + e^{-u}}$ to squeeze the result of linear regression to lie between 0 and 1. Interpret as a probability

$$p(y = 1 | \boldsymbol{x}, \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$$

- Can use maximum likelihood estimation to determine parameters $\boldsymbol{w}$. But there is no analytic solution because of nonlinearity.

Stochastic gradient descent

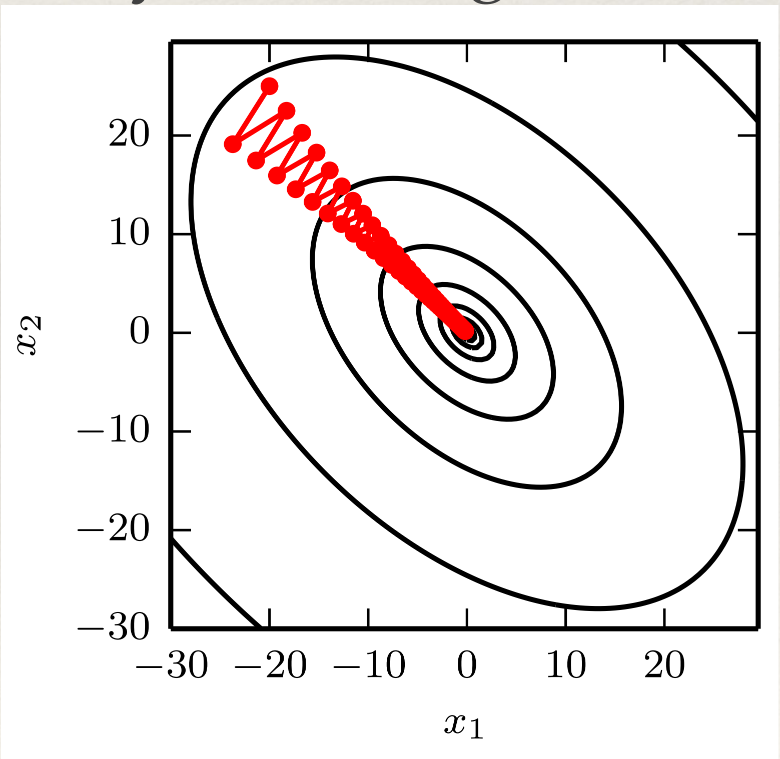
- In the case where a closed-form minimum is not available, **gradient descent** can be used to **optimize** the loss function, i.e., to tune $\boldsymbol{\theta}$ to approach the minimum.

- Starting from a point $\boldsymbol{\theta}_0$ we can move to a new point by following the gradient

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 - \epsilon \nabla_{\boldsymbol{\theta}} J|_{\boldsymbol{\theta}_0}$$

"Learning rate"

- Higher order algorithms can involve the second or higher derivatives (e.g., Hessian).



Goodfellow et al (2016)

Stochastic gradient descent

- For the negative log likelihood loss, the gradient reduces to the sum of per-example gradients,

$$\nabla_{\boldsymbol{\theta}} J = -\frac{1}{N} \sum_{i=1}^{N} \log p(y^{(i)} | \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$$

Cost $\propto N$

- Can break this up into **minibatches** (subsets of the full training set). Typically this could be several hundred training elements.
- This has two main advantages: (1) it is faster to compute each update, and (2) it introduces stochasticity, which helps avoid local minima.

Summary

- A machine learning algorithm requires the following:
  1. **dataset** — $\{\boldsymbol{x}^{(i)}, y^{(i)}\}$ (supervised) or $\{\boldsymbol{x}^{(i)}\}$ (unsupervised)
  2. **model** — E.g., linear regression $p_{\text{model}}(y \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\theta}^\top \boldsymbol{x}, 1)(y)$
  3. **loss function** — E.g., $J(\theta) = -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \log p_{\text{model}}(\boldsymbol{x})$
  4. **optimization algorithm** — E.g., stochastic gradient descent

Next lecture: deep learning

- **Challenges:**
  - High dimensionality of data:

    The number of possible data configurations is exponential in the number of data dimensions. Hard to cover this with training data.
  - Manifold learning:

    For many data sets, actual data realizations form a much lower dimensional subset of $\mathbb{R}^n$. E.g., random realizations of images will look like noise.