Statistical Machine Learning: Introduction

Dino Sejdinovic
Department of Statistics
University of Oxford

22-24 June 2015, Novi Sad
slides available at:
http://www.stats.ox.ac.uk/~sejdinov/talks.html

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to **learn** without being explicitly programmed.

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to **learn** without being explicitly programmed.

Tom Mitchell, 1997

Any computer program that **improves its performance** at some task **through experience**.

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to **learn** without being explicitly programmed.

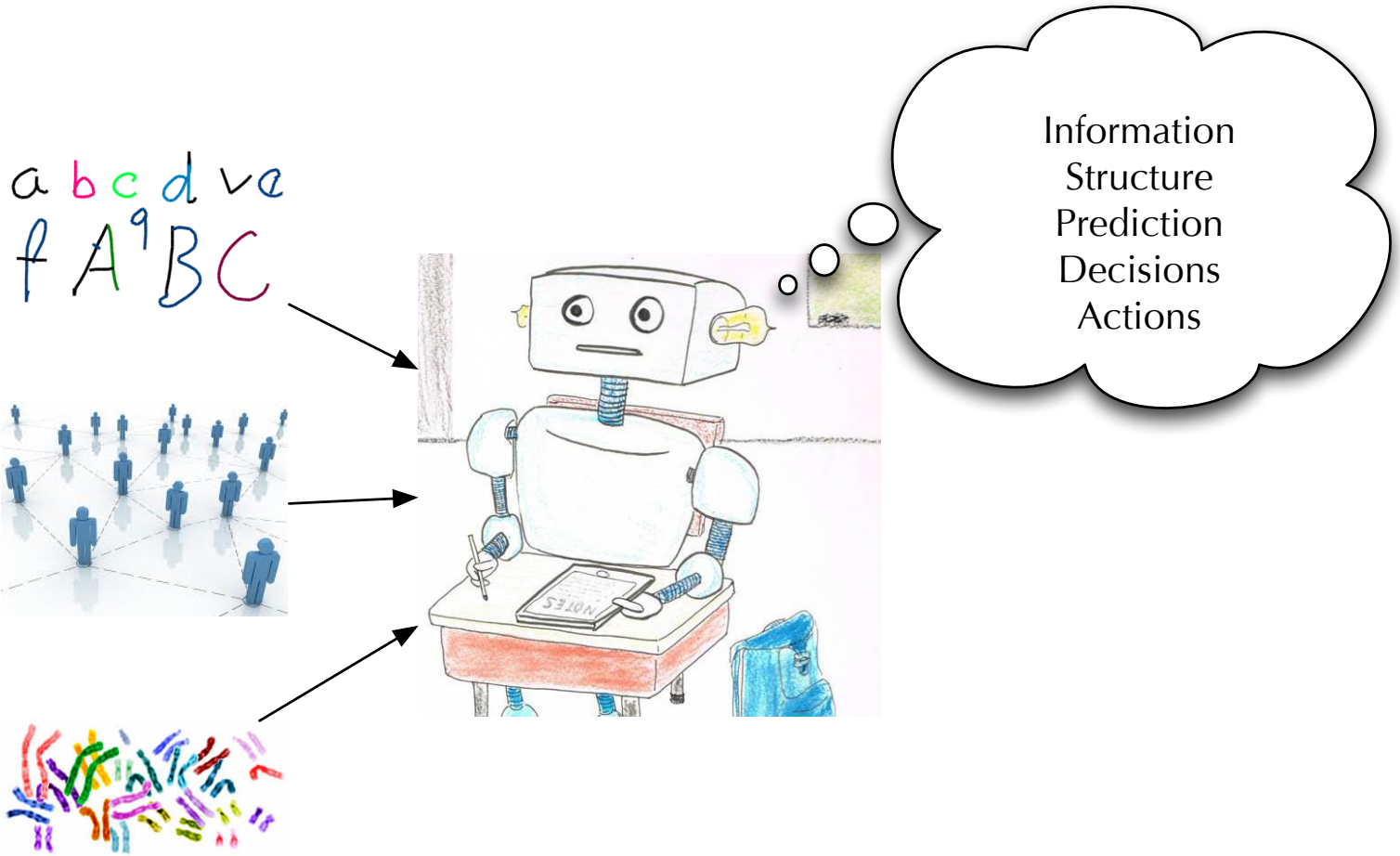
Tom Mitchell, 1997

Any computer program that **improves its performance** at some task **through experience**.

Kevin Murphy, 2012

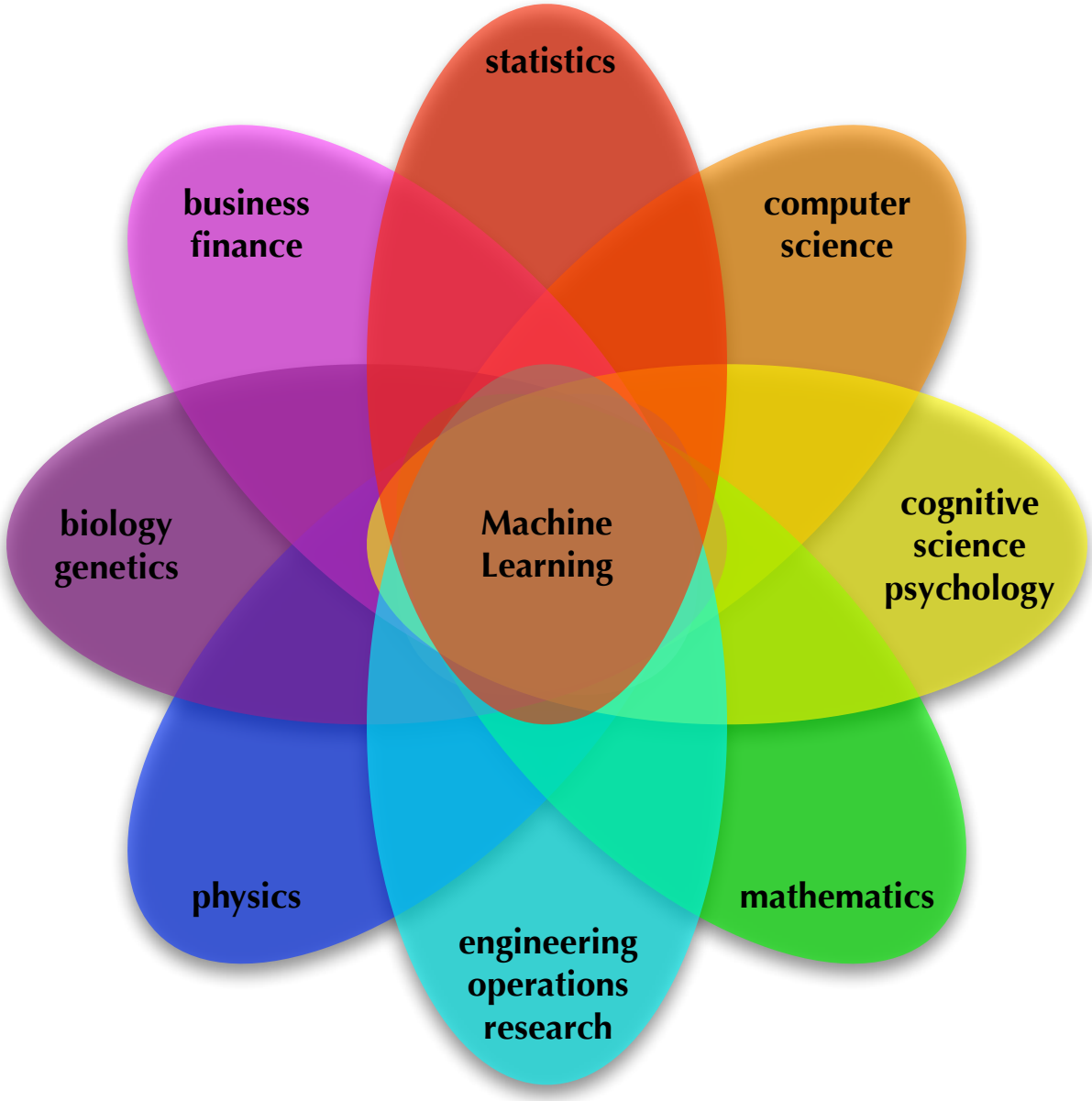
To develop methods that can **automatically** detect **patterns in data**, and then to use the uncovered patterns to **predict** future data or other outcomes of interest.

What is Machine Learning?

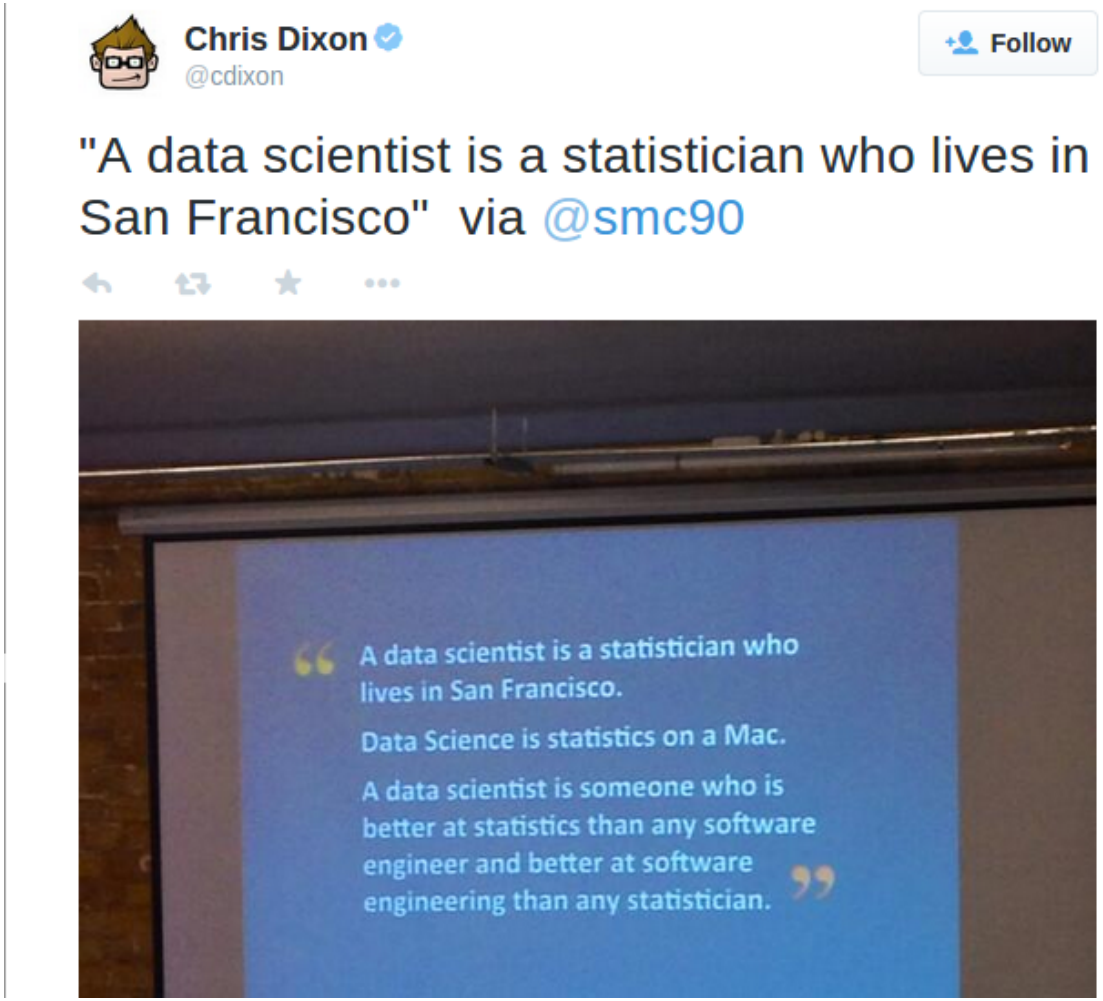


Larry Page about DeepMind's ML systems that can learn to play video games like humans

What is Machine Learning?



What is Data Science?



'Data Scientists' Meld Statistics and Software WSJ article

Information Revolution

Traditional Statistical Inference Problems

- Well formulated question that we would like to answer.
- Expensive data gathering and/or expensive computation.
- Create specially designed experiments to collect high quality data.

Information Revolution

- Improvements in data processing and data storage.
- Powerful, cheap, easy data capturing.
- Lots of (low quality) data with **potentially valuable** information inside.

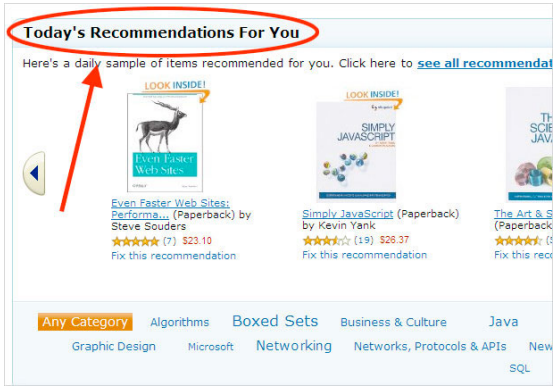
Statistics and Machine Learning in the age of Big Data

- ML becoming a thorough blending of computer science, engineering and statistics
- unified framework of data, inferences, procedures, algorithms
  - statistics taking computation seriously
  - computing taking statistical risk seriously
- scale and granularity of data
- personalization, societal and business impact
- multidisciplinarity - and you are the interdisciplinary glue
- it's just getting started

Michael Jordan: On the Computational and Statistical Interface and "Big Data"

Applications of Machine Learning

spam filtering



recommendation systems

fraud detection

self-driving cars

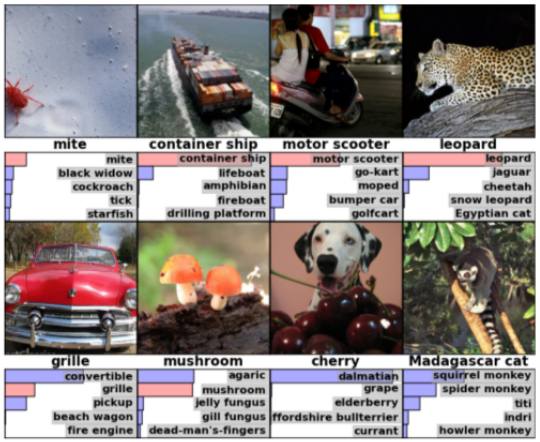


image recognition

stock market analysis

ImageNet: Krizhevsky et al, 2012; Machine Learning is Eating the World: Forbes article

Types of Machine Learning

Supervised learning

- Data contains “labels”: every example is an input-output pair
- classification, regression
- Goal: **prediction on new examples**

Unsupervised learning

- Extract key features of the “unlabelled” data
- clustering, signal separation, density estimation
- Goal: **representation, hypothesis generation, visualization**

Types of Machine Learning

Semi-supervised Learning

A database of examples, only a small subset of which are labelled.

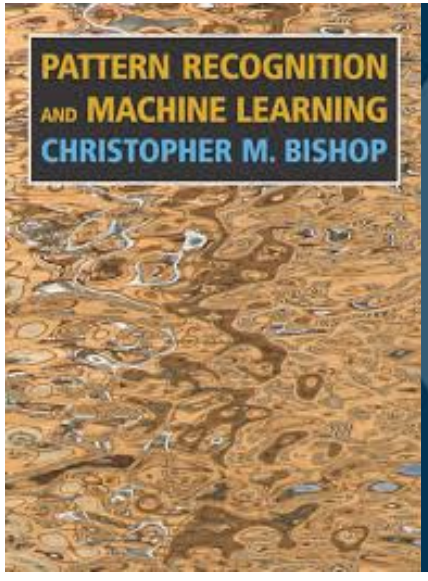
Multi-task Learning

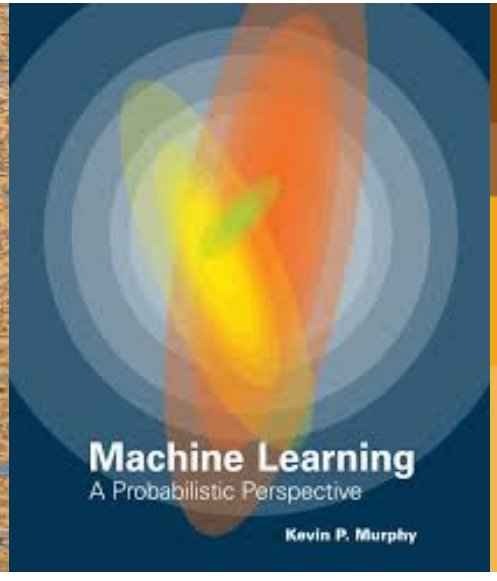
A database of examples, each of which has multiple labels corresponding to different prediction tasks.

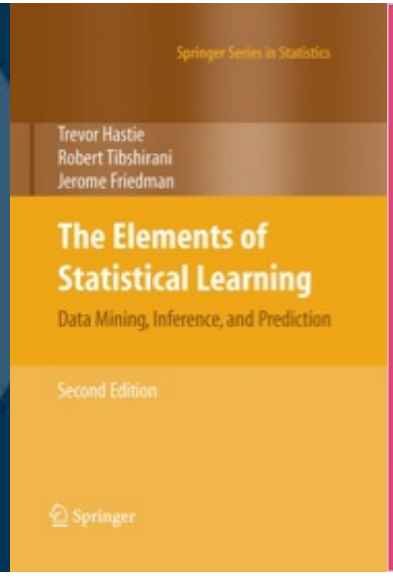
Reinforcement Learning

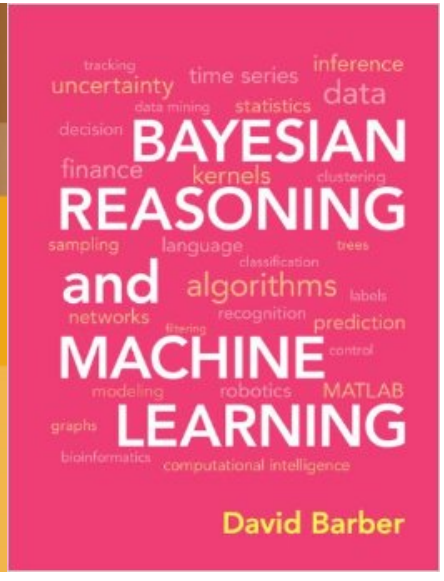
An agent acting in an environment, given rewards for performing appropriate actions, learns to maximize their reward.

Literature


PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP


Machine Learning
A Probabilistic Perspective
Kevin P. Murphy


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer


tracking
uncertainty
time series
inference
data mining
statistics
data
decision
BAYESIAN
finance
kernels
clustering
REASONING
sampling
language
trees
classification
and
algorithms
labels
networks
recognition
prediction
filtering
MACHINE
control
modeling
robotics
MATLAB
graphs
LEARNING
bioinformatics
computational intelligence
David Barber

Supervised Learning

- We observe the input-output pairs $\{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$
- Types of supervised learning:
  - Classification: discrete responses, e.g. $\mathcal{Y} = \{+1, -1\}$ or $\{1, \ldots, K\}$.
  - Regression: a numerical value is observed and $\mathcal{Y} = \mathbb{R}$.

The goal is to accurately predict the response Y on new observations of X, i.e., to **learn a function** $f : \mathbb{R}^p \to \mathcal{Y}$, such that $f(X)$ will be close to the true response Y.

Loss function

- Suppose we made a prediction $\hat{Y} = f(X) \in \mathcal{Y}$ based on observation of X.
- How good is the prediction? We can use a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ to formalize the quality of the prediction.
- Typical loss functions:
  - **Misclassification loss** (or **0-1 loss**) for classification

$$L(Y, f(X)) = \begin{cases} 0 & f(X) = Y \\ 1 & f(X) \neq Y \end{cases}.$$

  - **Squared loss** for regression

$$L(Y, f(X)) = (f(X) - Y)^2.$$

- Many other choices are possible, e.g., **weighted misclassification loss**.
- In classification, the vector of estimated probabilities $(\hat{p}(k))_{k \in \mathcal{Y}}$ can be returned, and in this case **log-likelihood loss** (or **log loss**) $L(Y, \hat{p}) = -\log \hat{p}(Y)$ is often used.

Risk

- Paired observations $\{(x_i, y_i)\}_{i=1}^n$ viewed as i.i.d. realizations of a random variable (X, Y) on $\mathcal{X} \times \mathcal{Y}$ with joint distribution P_{XY}

Risk

For a given loss function L, the **risk** R of a learned function f is given by the expected loss

$$R(f) = \mathbb{E}_{P_{XY}}\left[L(Y, f(X))\right],$$

where the expectation is with respect to the true (unknown) joint distribution of (X, Y).

- The risk is unknown, but we can compute the **empirical risk**:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)).$$

The Bayes Classifier

- What is the optimal classifier if the joint distribution (X, Y) were known?
- The density g of X can be written as a mixture of K components (corresponding to each of the classes):

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(x),$$

where, for $k = 1, \ldots, K$,

  - $\mathbb{P}(Y = k) = \pi_k$ are the class probabilities,
  - $g_k(x)$ is the conditional density of X, given $Y = k$.

- The **Bayes classifier** $f_{\text{Bayes}} : x \mapsto \{1, \ldots, K\}$ is the one with minimum risk:

$$\begin{aligned} R(f) =& \mathbb{E}\left[L(Y, f(X))\right] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[L(Y, f(X))|X]\right] \\ =& \int_{\mathcal{X}} \mathbb{E}\left[L(Y, f(X))|X = x\right] g(x) dx \end{aligned}$$

- The minimum risk attained by the Bayes classifier is called **Bayes risk**.
- Minimizing $\mathbb{E}[L(Y, f(X))|X = x]$ separately for each x suffices.

The Bayes Classifier

- Consider the 0-1 loss.
- The risk simplifies to:

$$\begin{aligned}\mathbb{E}\Big[L(Y,f(X))\Big|X=x\Big] &= \sum_{k=1}^{K} L(k,f(x))\mathbb{P}(Y=k|X=x)\\ &= 1-\mathbb{P}(Y=f(x)|X=x)\end{aligned}$$

- The risk is minimized by choosing the class with the greatest posterior probability:

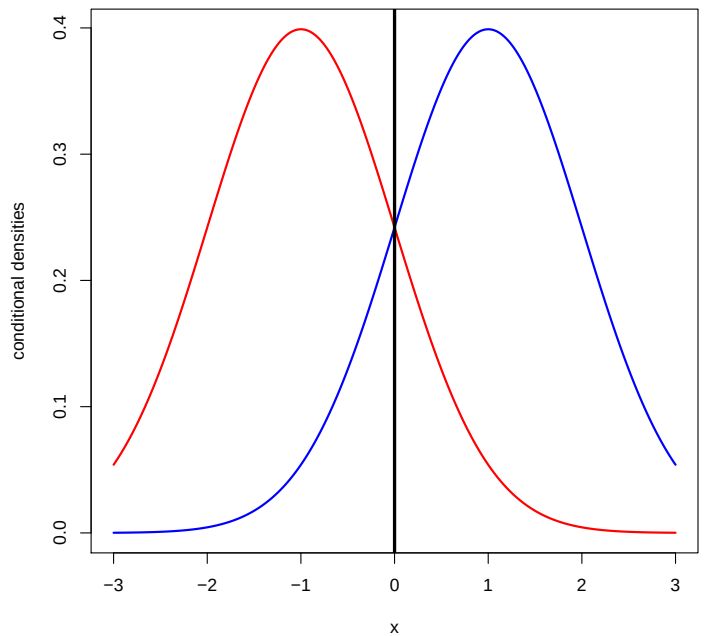
$$\begin{aligned}f_{\text{Bayes}}(x) &= \underset{k=1,\ldots,K}{\arg\max}\,\mathbb{P}(Y=k|X=x)\\ &= \underset{k=1,\ldots,K}{\arg\max}\,\frac{\pi_k g_k(x)}{\sum_{j=1}^{K}\pi_j g_j(x)} = \underset{k=1,\ldots,K}{\arg\max}\;\pi_k g_k(x).\end{aligned}$$

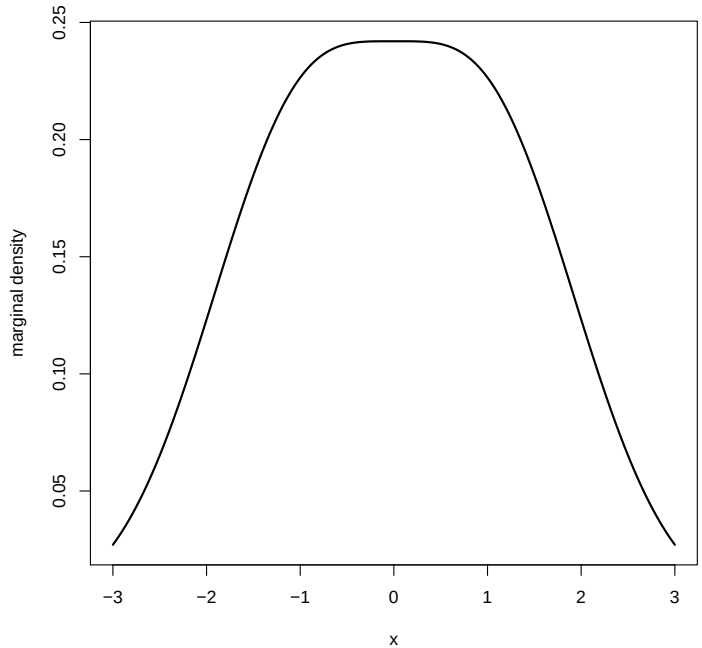
- The functions $x \mapsto \pi_k g_k(x)$ are called **discriminant functions**. The discriminant function with maximum value determines the predicted class of x.

The Bayes Classifier: Example

A simple two Gaussians example: Suppose $X \sim \mathcal{N}(\mu_Y, 1)$, where $\mu_1 = -1$ and $\mu_2 = 1$ and assume equal priors $\pi_1 = \pi_2 = 1/2$.

$$g_1(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x+1)^2}{2}\right) \quad \text{and} \quad g_2(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right).$$

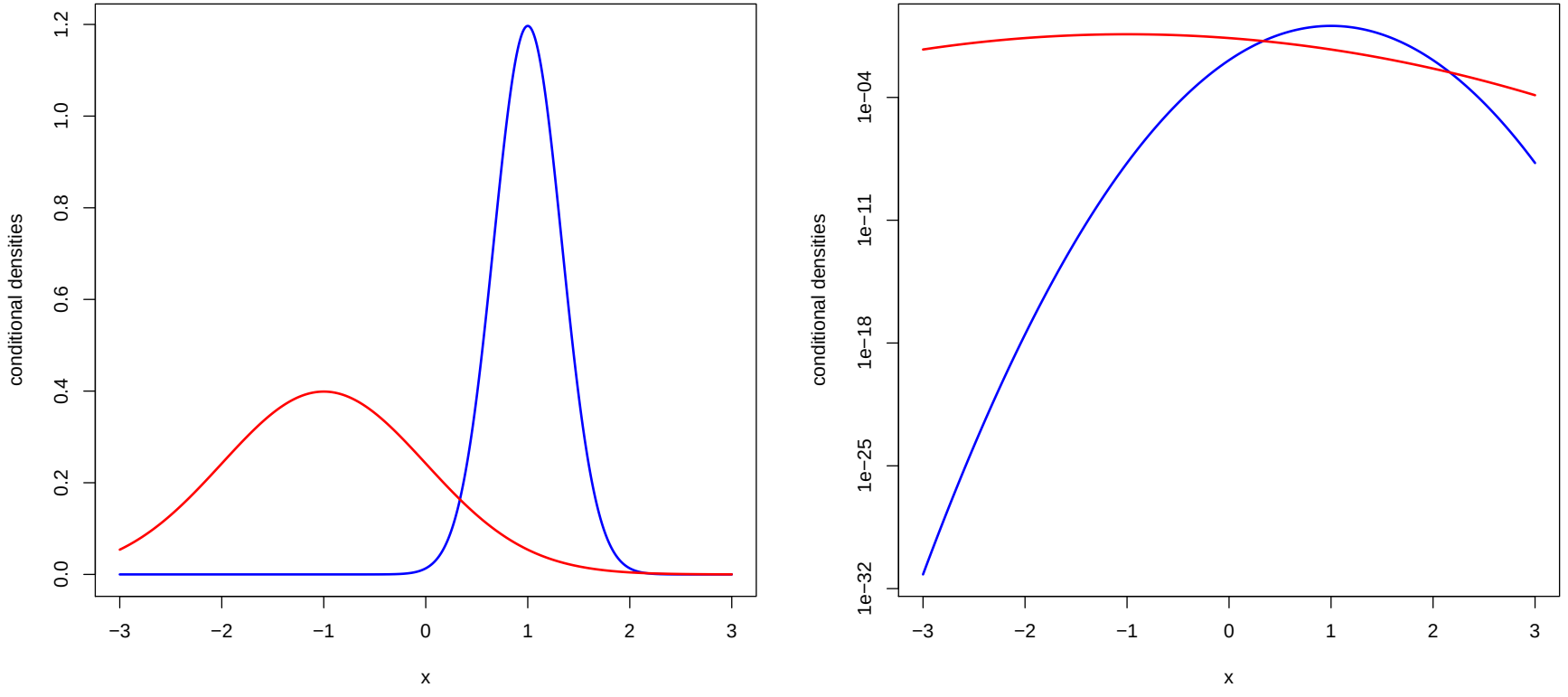




Optimal classification is $f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\arg\max}\ \pi_k g_k(x) = \begin{cases} 1 & \text{if } x < 0, \\ 2 & \text{if } x \geq 0. \end{cases}$

The Bayes Classifier: Example

How do you classify a new observation x if now the standard deviation is still 1 for class 1 but $1/3$ for class 2?



Looking at density in a log-scale, optimal classification is to select class 2 if and only if $x \in [0.34, 2.16]$.

Plug-in Classification

- The Bayes Classifier chooses the class with the greatest posterior probability

$$f_{\text{Bayes}}(x) \quad = \quad \underset{k=1,\ldots,K}{\arg\max}\, \pi_k g_k(x).$$

- We know neither the conditional densities g_k nor the class probabilities π_k!
- The **plug-in classifier** chooses the class

$$f(x) = \underset{k=1,\ldots,K}{\arg\max}\, \hat{\pi}_k \hat{g}_k(x),$$

- where we plugged in
  - estimates $\hat{\pi}_k$ of π_k and $k = 1, \ldots, K$ and
  - estimates $\hat{g}_k(x)$ of conditional densities,

Linear Discriminant Analysis

- **LDA** is the simplest example of plug-in classification.
- Assume multivariate normal conditional density $g_k(x)$ for each class k:

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

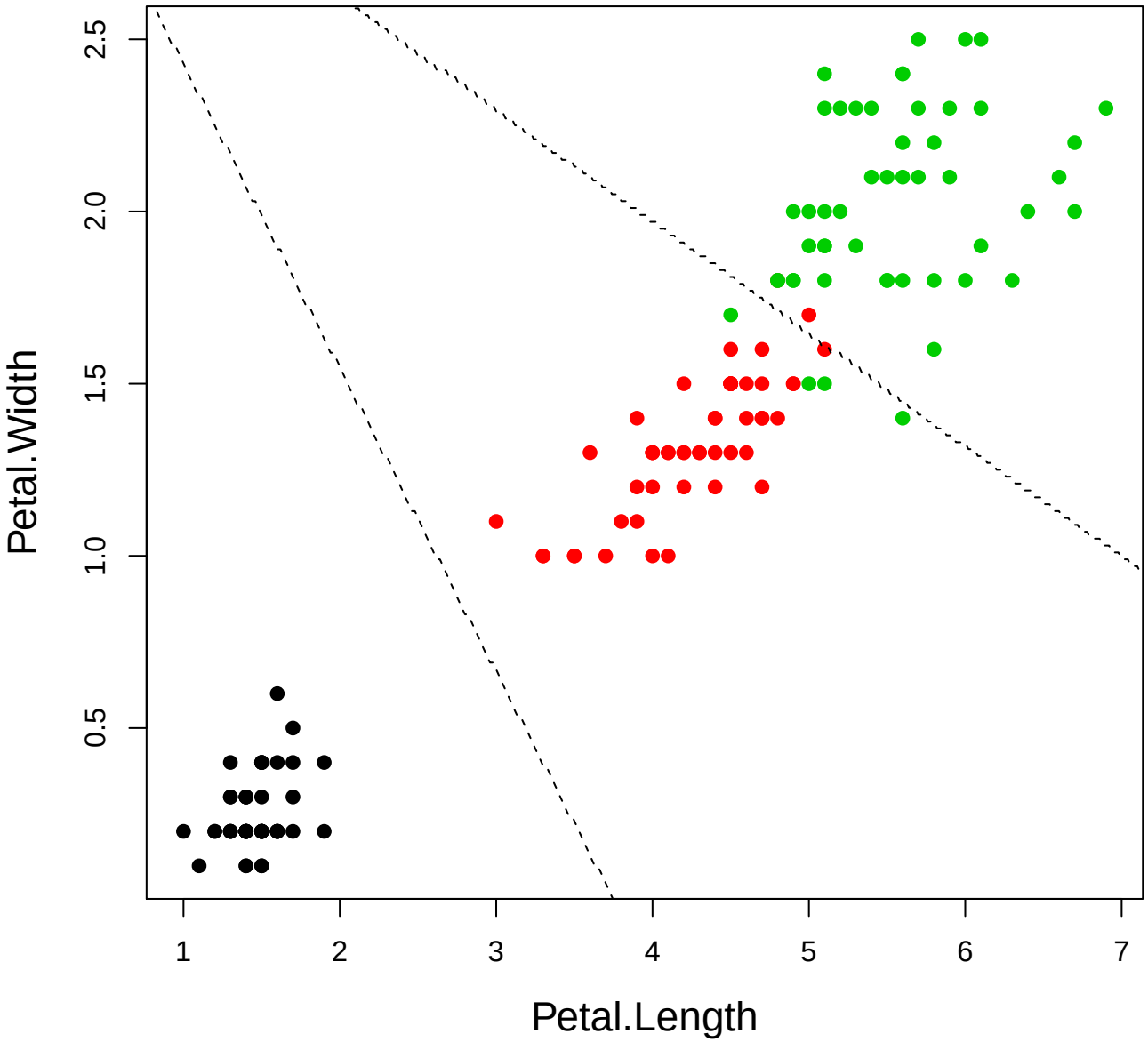
$$g_k(x) = (2\pi)^{-p/2}|\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)\right),$$

  - each class can have a **different mean** μ_k,
  - all classes share the **same covariance** Σ.
- For an observation x, the k-th log-discriminant function is

$$\begin{aligned} \log \pi_k g_k(x) &= \text{const} + \log \pi_k - \frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k) \\ &= \text{const} + a_k + b_k^\top x \end{aligned}$$

  linear log-discriminant functions $\leftrightarrow$ linear decision boundaries.
- The quantity $(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k)$ is the squared **Mahalanobis distance** between x and μ_k.
- If $\Sigma = I_p$ and $\pi_k = \frac{1}{K}$, LDA simply chooses the class k with the nearest (in the Euclidean sense) class mean.

Iris Dataset



Generative vs Discriminative Learning

- **Generative learning**: find parameters which **explain all the data available**.

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log P(x_i, y_i|\theta)$$

Examples: LDA, naïve Bayes.

  - Makes use of all the data available.
  - Flexible framework, can incorporate other tasks, incomplete data.
  - Stronger modelling assumptions.

- **Discriminative learning**: find parameters that aid in **prediction**.

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i)) \quad \text{or} \quad \hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log P(y_i|x_i, \theta)$$

Examples: logistic regression, support vector machines.

  - Typically performs better on a given task.
  - Weaker modelling assumptions.
  - Can overfit more easily.

Generative Learning

- We work with a joint distribution $P_{X,Y}(x,y)$ over data vectors and labels.
- A learning algorithm: construct $f : \mathcal{X} \to \mathcal{Y}$ which predicts the label of X.
- Given a loss function L, the risk R of $f(X)$ is

$$R(f) = \mathbb{E}_{X,Y}[L(Y, f(X))]$$

- For 0/1 loss in classification, Bayes classifier

$$f_{\text{Bayes}}(x) = \operatorname*{argmax}_{k=1,\ldots,K} P(Y = k|x) = \operatorname*{argmax}_{k=1,\ldots,K} P_{X,Y}(x,k)$$

  has the minimum risk (Bayes risk), but is unknown since $P_{X,Y}$ is unknown.
- Assume a parameteric model for the joint: $P_{X,Y}(x,y) = P_{X,Y}(x,y|\theta)$
- Fit $\hat{\theta} = \operatorname{argmax}_\theta \sum_{i=1}^n \log P(x_i, y_i|\theta)$ and plug in back to Bayes classifier:

$$\hat{f}(x) = \operatorname*{argmax}_{k=1,\ldots,K} P_{X,Y}(x,k|\hat{\theta}).$$

Hypothesis space and Empirical Risk Minimization

- Find best function in $\mathcal{H}$ minimizing the risk:

$$f_\star = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{X,Y}[L(Y, f(X))]$$

- **Empirical Risk Minimization** (ERM): minimize the empirical risk instead, since we typically do not know $P_{X,Y}$.

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))$$

- Hypothesis space $\mathcal{H}$ is the space of functions f under consideration.
- How complex should we allow functions f to be? If hypothesis space $\mathcal{H}$ is "too large", ERM will overfit. Function

$$\hat{f}(x) = \begin{cases} y_i & \text{if } x = x_i, \\ 0 & \text{otherwise} \end{cases}$$

will have zero empirical risk, but is useless for generalization, since it has simply "memorized" the dataset.

Training and Test Performance

- **Training error** is the empirical risk

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$$

  For 0-1 loss in classification, this is the misclassification error on the training data $\{x_i, y_i\}_{i=1}^{n}$, **which were used in learning** f.

- **Test error** is the empirical risk on **new, previously unseen** observations $\{\tilde{x}_i, \tilde{y}_i\}_{i=1}^{m}$

$$\frac{1}{m}\sum_{i=1}^{m} L(\tilde{y}_i, f(\tilde{x}_i))$$

  which were NOT used in learning f.

- Test error is a much better gauge of how well learned function **generalizes** to new data.
- The test error is in general larger than the training error.

Hypothesis space for two-class LDA

- Assume we have two classes $\{+1, -1\}$.
- Recall that the discriminant functions in LDA are linear. Assuming that data vectors in class k is modelled as $\mathcal{N}(\mu_k, \Sigma)$, choosing class $+1$ over -1 involves:

$$a_{+1} + b_{+1}^\top x > a_{-1} + b_{-1}^\top x \qquad \Leftrightarrow \qquad a_\star + b_\star^\top x > 0,$$

  where $a_\star = a_{+1} - a_{-1}$, $b_\star = b_{+1} - b_{-1}$.
- Thus, hypothesis space of two-class LDA consists of functions $f(x) = \text{sign}(a + b^\top x)$.
- We obtain coefficients $\hat{a}$ and $\hat{b}$, and thus the function $\hat{f}$ through fitting the parameters of the generative model.
- **Discriminative learning**: restrict $\mathcal{H}$ to a class of functions $f(x) = \text{sign}(a + b^\top x)$ and select $\hat{a}$ and $\hat{b}$ which minimize empirical risk.

Space of linear decision functions

- Hypothesis space $\mathcal{H} = \{f : f(x) = \text{sign}(a + b^\top x), a \in \mathbb{R}, b \in \mathbb{R}^p\}$
- Find a, b that minimize the empirical risk under 0-1 loss:

$$\begin{aligned}
&\underset{a,b}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_{a,b}(x_i)) \\
&= \underset{a,b}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0 & \text{if } y_i = \text{sign}(a + b^\top x_i) \\ 1 & \text{otherwise} \end{cases} \\
&= \underset{a,b}{\text{argmin}} \frac{1}{2n} \sum_{i=1}^{n} \left[1 - \text{sign}(y_i(a + b^\top x_i))\right].
\end{aligned}$$

- Combinatorial problem - not typically possible to solve...
- Maybe easier with a different loss function? (Logistic regression)

Linearity of log-posterior odds

- Another way to express linear decision boundary of LDA:

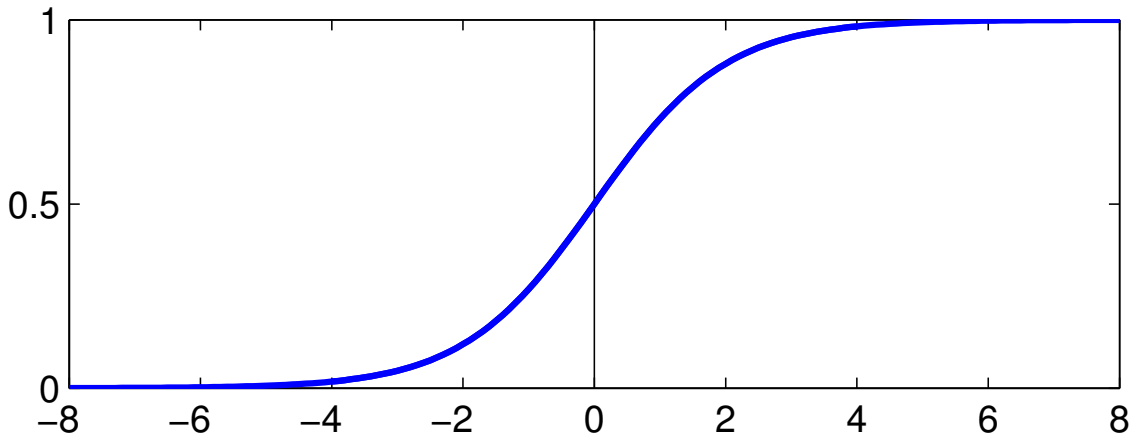
$$\log \frac{p(Y=+1|X=x)}{p(Y=-1|X=x)} = a + b^\top x.$$

- Solve explicitly for conditional class probabilities:

$$p(Y=+1|X=x) = \frac{1}{1+\exp(-(a+b^\top x))} =: s(a+b^\top x)$$

$$p(Y=-1|X=x) = \frac{1}{1+\exp(+(a+b^\top x))} = s(-a-b^\top x)$$

where $s(\cdot)$ is the **logistic function**



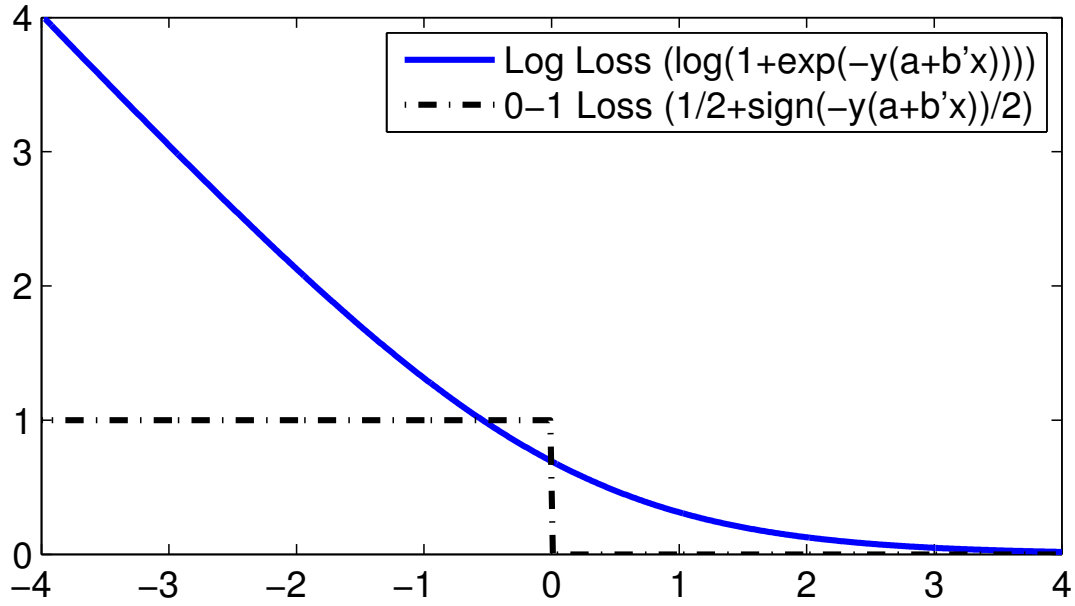
Logistic Regression

- Consider maximizing the **conditional log likelihood**:

$$\ell(a,b) = \sum_{i=1}^{n} \log p(Y = y_i | X = x_i) = \sum_{i=1}^{n} -\log(1 + \exp(-y_i(a + b^\top x_i)))$$

- Equivalent to minimizing the empirical risk associated with the **log loss**:

$$R_{\log}^{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i(a + b^\top x_i))) = \frac{1}{n} \sum_{i=1}^{n} \left[-\log(\underbrace{s(y_i(a + b^\top x_i))}_{\hat{p}_{a,b}(y_i|x_i)}) \right]$$



Logistic Regression

- Not possible to find optimal a, b analytically.
- For simplicitiy, absorb a as an entry in b by appending '1' into x vector.
- Objective function:

$$R^{\text{emp}}_{\log} = \frac{1}{n}\sum_{i=1}^{n} -\log s(y_i x_i^\top b)$$

Logistic Function

$$s(-z) = 1 - s(z)$$
$$\nabla_z s(z) = s(z)s(-z)$$
$$\nabla_z \log s(z) = s(-z)$$
$$\nabla_z^2 \log s(z) = -s(z)s(-z)$$

- Differentiate wrt b:

$$\nabla_b R^{\text{emp}}_{\log} = \frac{1}{n}\sum_{i=1}^{n} -s(-y_i x_i^\top b) y_i x_i$$

$$\nabla_b^2 R^{\text{emp}}_{\log} = \frac{1}{n}\sum_{i=1}^{n} s(y_i x_i^\top b)s(-y_i x_i^\top b) x_i x_i^\top$$

Logistic Regression

- Second derivative is positive-definite: objective function is **convex** and there is **a single unique global minimum**.
- Many different algorithms can find optimal b, e.g.:
  - Gradient descent:

$$b^{\text{new}} = b + \epsilon \frac{1}{n} \sum_{i=1}^{n} s(-y_i x_i^\top b) y_i x_i$$

  - Stochastic gradient descent:

$$b^{\text{new}} = b + \epsilon_t \frac{1}{|I(t)|} \sum_{i \in I(t)} s(-y_i x_i^\top b) y_i x_i$$

    where $I(t)$ is a subset of the data at iteration t, and $\epsilon_t \to 0$ slowly ($\sum_t \epsilon_t = \infty, \sum_t \epsilon_t^2 < \infty$).
  - Newton-Raphson:

$$b^{\text{new}} = b - (\nabla_b^2 R_{\log}^{\text{emp}})^{-1} \nabla_b R_{\log}^{\text{emp}}$$

    This is also called **iterative reweighted least squares**.
  - Conjugate gradient, LBFGS and other methods from numerical analysis.

Logistic Regression vs. LDA

- Both have linear decision boundaries and model log-posterior odds as

$$\log \frac{p(Y=+1|X=x)}{p(Y=-1|X=x)} = a + b^\top x$$

- LDA models the marginal density of x as a Gaussian mixture with shared covariance

$$g(x) = \pi_{-1}\mathcal{N}(x;\mu_{-1},\Sigma) + \pi_{+1}\mathcal{N}(x;\mu_{+1},\Sigma)$$

  and fits the parameters $\theta = (\mu_{-1},\mu_{+1},\pi_{-1},\pi_{+1},\Sigma)$ by maximizing joint likelihood $\sum_{i=1}^n p(x_i,y_i|\theta)$. a and b are then determined from θ.
- Logistic regression leaves the marginal density $g(x)$ as an **arbitrary density function**, and fits the parameters a,b by maximizing the conditional likelihood $\sum_{i=1}^n p(y_i|x_i;a,b)$.

Logistic Regression

Properties of logistic regression:

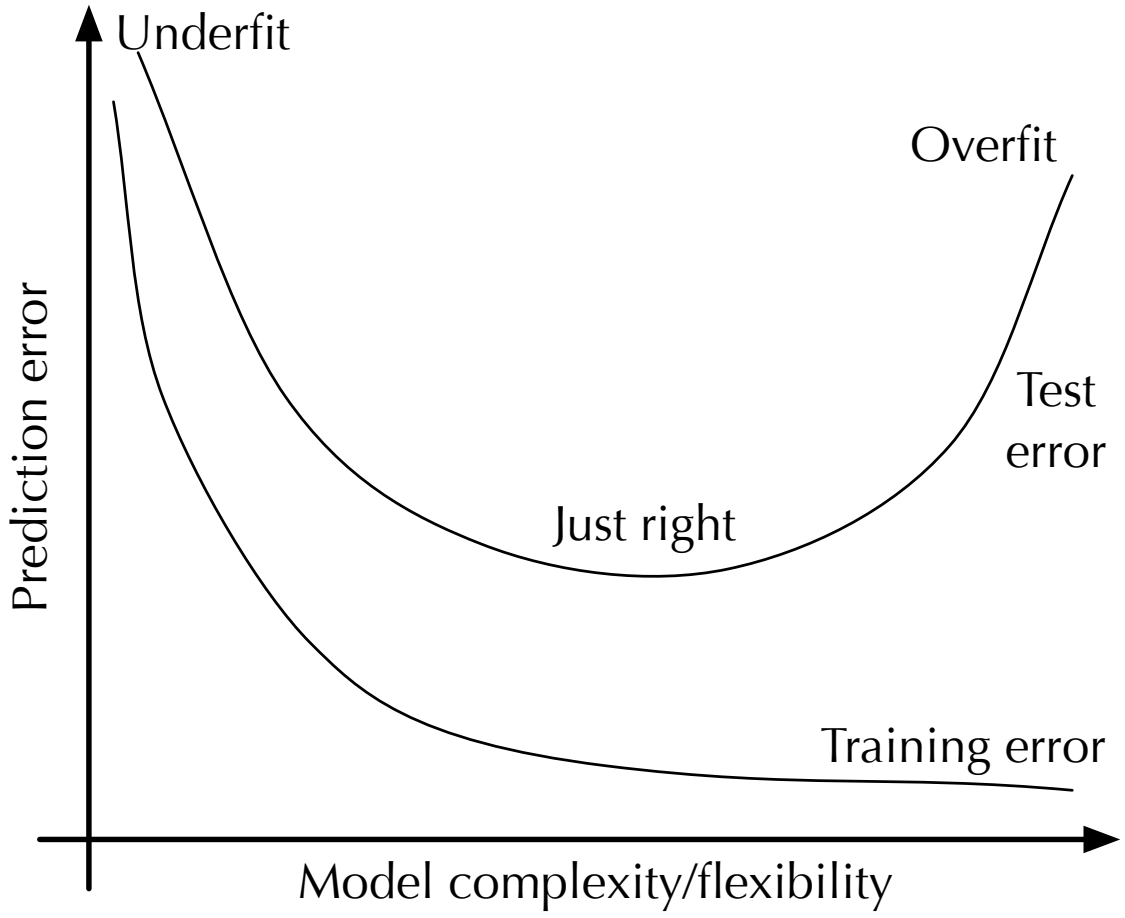
- Makes less modelling assumptions than generative classifiers.
- A simple example of a generalised linear model (GLM). Much statistical theory:
  - assessment of fit via deviance and plots,
  - interpretation of entries of b as **odds-ratios**,
  - fitting categorical data (sometimes called **multinomial logistic regression**),
  - well founded approaches to removing insignificant features (drop-in deviance test, Wald test).

Model Complexity and Generalization

Generalization

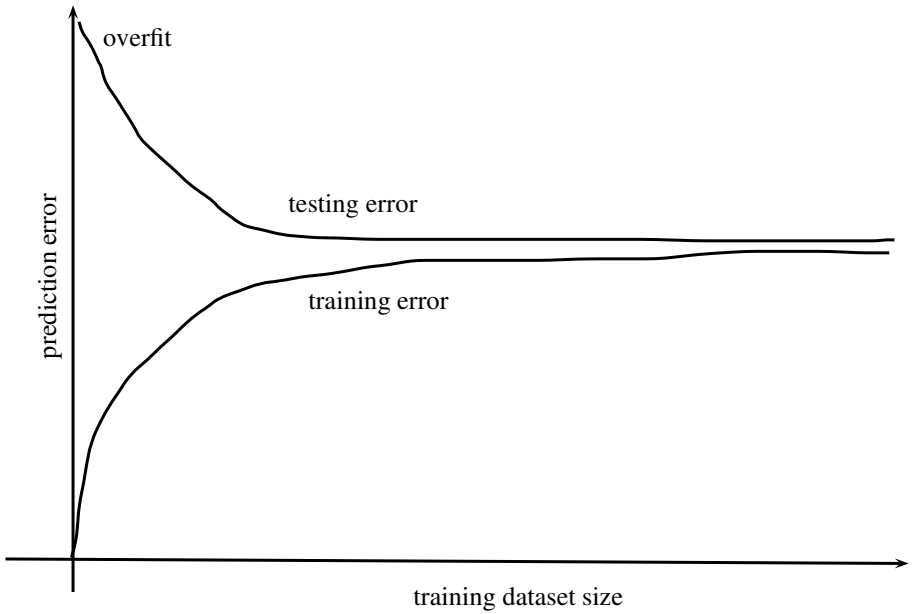
- Generalization ability: what is the out-of-sample error of learner f?
- training error $\neq$ testing error.
- We learn f by minimizing some variant of empirical risk $R^{\text{emp}}(f)$- what can we say about the true risk $R(f)$?
- Two important factors determining generalization ability:
  - Model complexity
  - Training data size

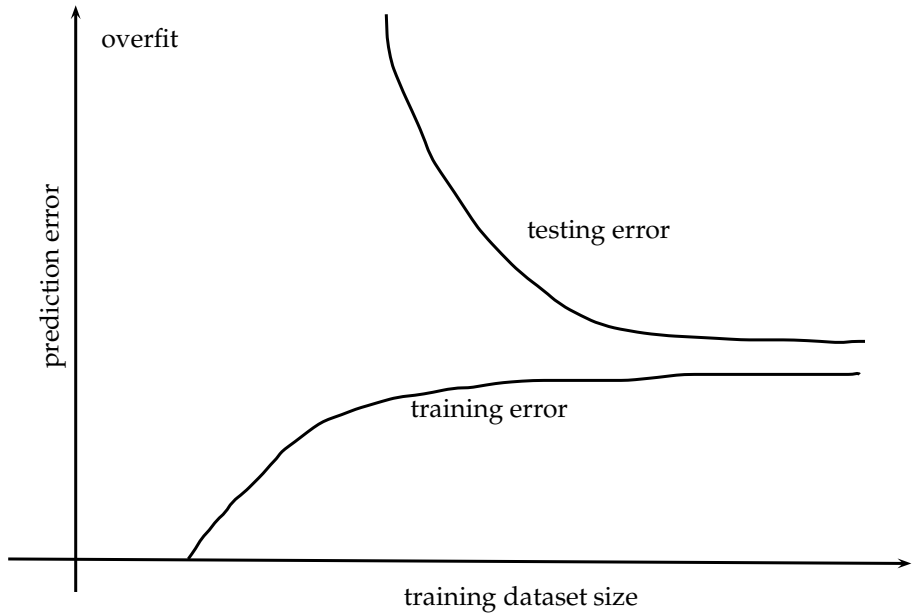
Learning Curves



Fixed dataset size, varying model complexity.

Learning Curves





Fixed model complexity, varying dataset size.
Two models: one simple, one complex. Which is which?

Bias-Variance Tradeoff

- Where does the prediction error come from?
- Example: Squared loss in regression: $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$,

$$L(Y, f(X)) = (Y - f(X))^2$$

- Optimal f is the conditional mean:

$$f_*(x) = \mathbb{E}[Y|X = x]$$

- Follows from $R(f) = \mathbb{E}_X \mathbb{E}\left[(Y - f(X))^2 \middle| X\right]$ and

$$\begin{aligned} & \mathbb{E}\left[(Y - f(X))^2 \middle| X = x\right] \\ = \; & \mathbb{E}\left[Y^2 \middle| X = x\right] - 2f(x)\,\mathbb{E}[Y|X = x] + f(x)^2 \\ = \; & \mathrm{Var}[Y|X = x] + (\mathbb{E}[Y|X = x] - f(x))^2 . \end{aligned}$$

Bias-Variance Tradeoff

- Optimal risk is the intrinsic conditional variability of Y (noise):

$$R(f_*) = \mathbb{E}_X \left[\mathrm{Var} \left[Y | X \right] \right]$$

- **Excess risk**:

$$R(f) - R(f_*) = \mathbb{E}_X \left[\left(f(X) - f_*(X) \right)^2 \right]$$

- How does the excess risk behave **on average**?
- Consider a randomly selected dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n$ and $f^{(\mathcal{D})}$ trained on $\mathcal{D}$ using a model (hypothesis class) $\mathcal{H}$.

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} \left[R(f^{(\mathcal{D})}) - R(f_*) \right] &= \mathbb{E}_{\mathcal{D}} \mathbb{E}_X \left[\left(f^{(\mathcal{D})}(X) - f_*(X) \right)^2 \right] \\ &= \mathbb{E}_X \mathbb{E}_{\mathcal{D}} \left[\left(f^{(\mathcal{D})}(X) - f_*(X) \right)^2 \right]. \end{aligned}$$

Bias-Variance Tradeoff

- Denote $\bar{f}(x) = \mathbb{E}_{\mathcal{D}} f^{(\mathcal{D})}(x)$ (average decision function over all possible datasets)

$$\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - f_*(X)\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - \bar{f}(X)\right)^2\right]}_{\mathrm{Var}_X(\mathcal{H},n)} + \underbrace{\left(\bar{f}(X) - f_*(X)\right)^2}_{\mathrm{Bias}^2_X(\mathcal{H},n)}$$

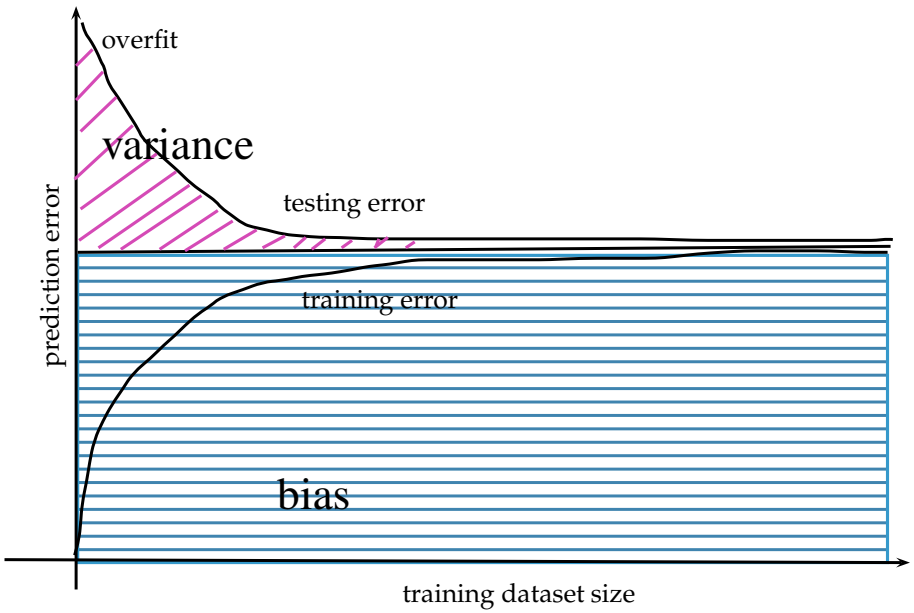
Now,

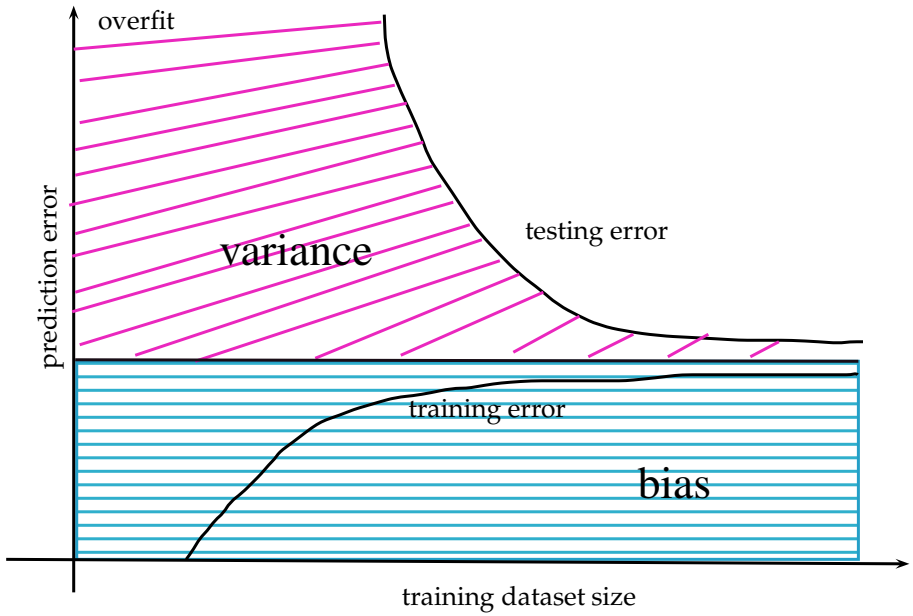
$$\mathbb{E}_{\mathcal{D}} R(f^{(\mathcal{D})}) = R(f_*) + \mathbb{E}_X \mathrm{Var}_X(\mathcal{H}, n) + \mathbb{E}_X Bias^2_X(\mathcal{H}, n)$$

Where does the prediction error come from?

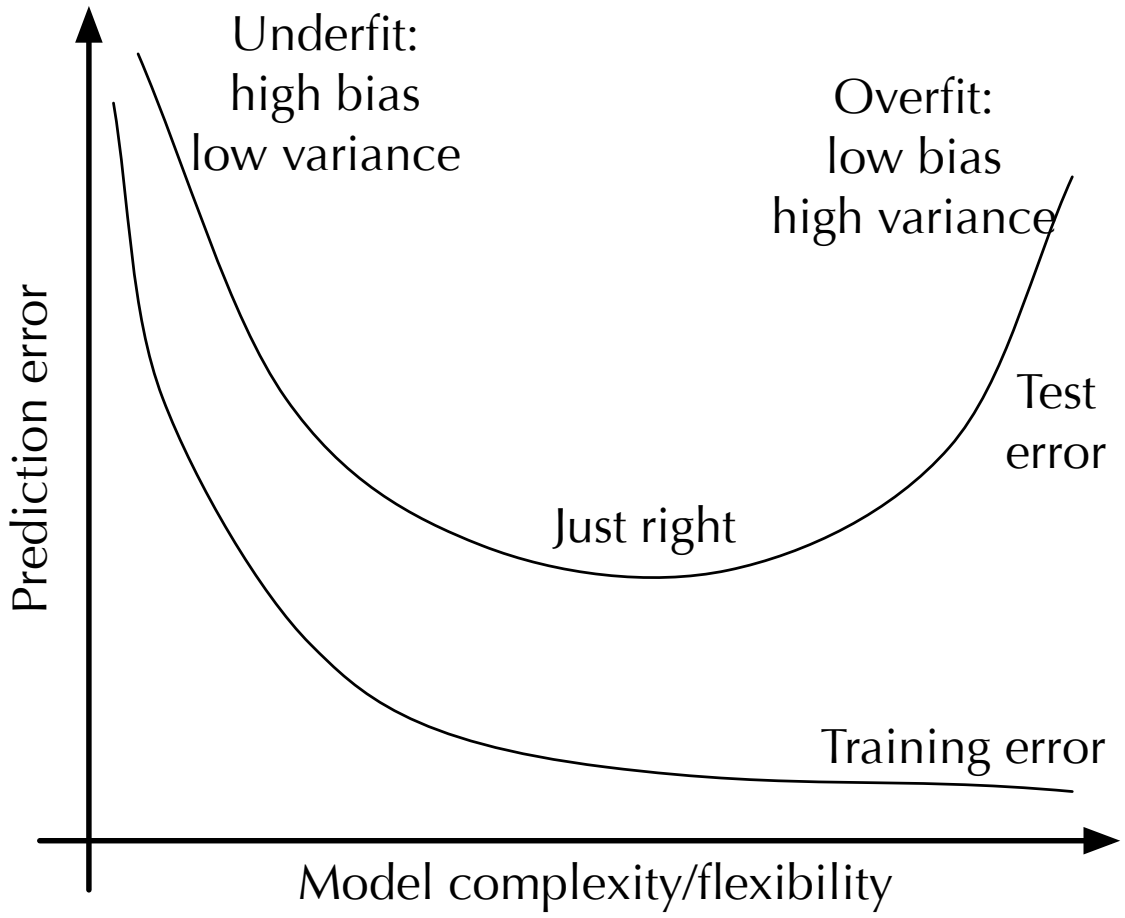
- **Noise**: Intrinsic difficulty of regression problem.
- **Bias**: How far away is the best learner in the model (average learner over all possible datasets) from the optimal one?
- **Variance**: How variable is our learning method if given different datasets?

Learning Curves





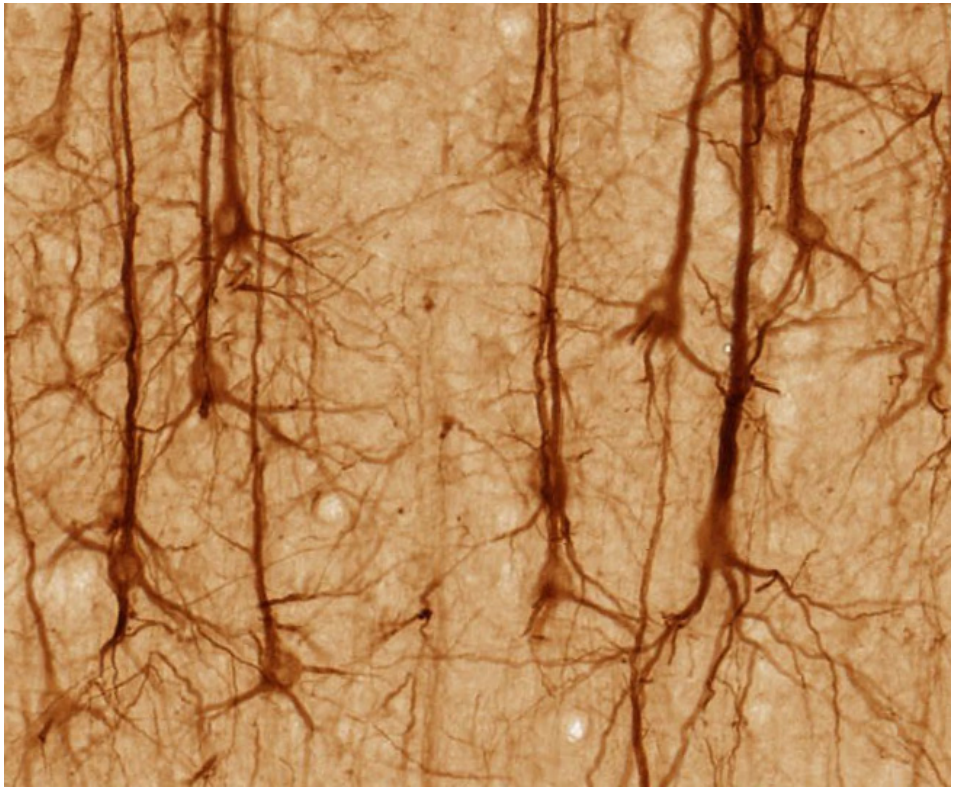
Learning Curve



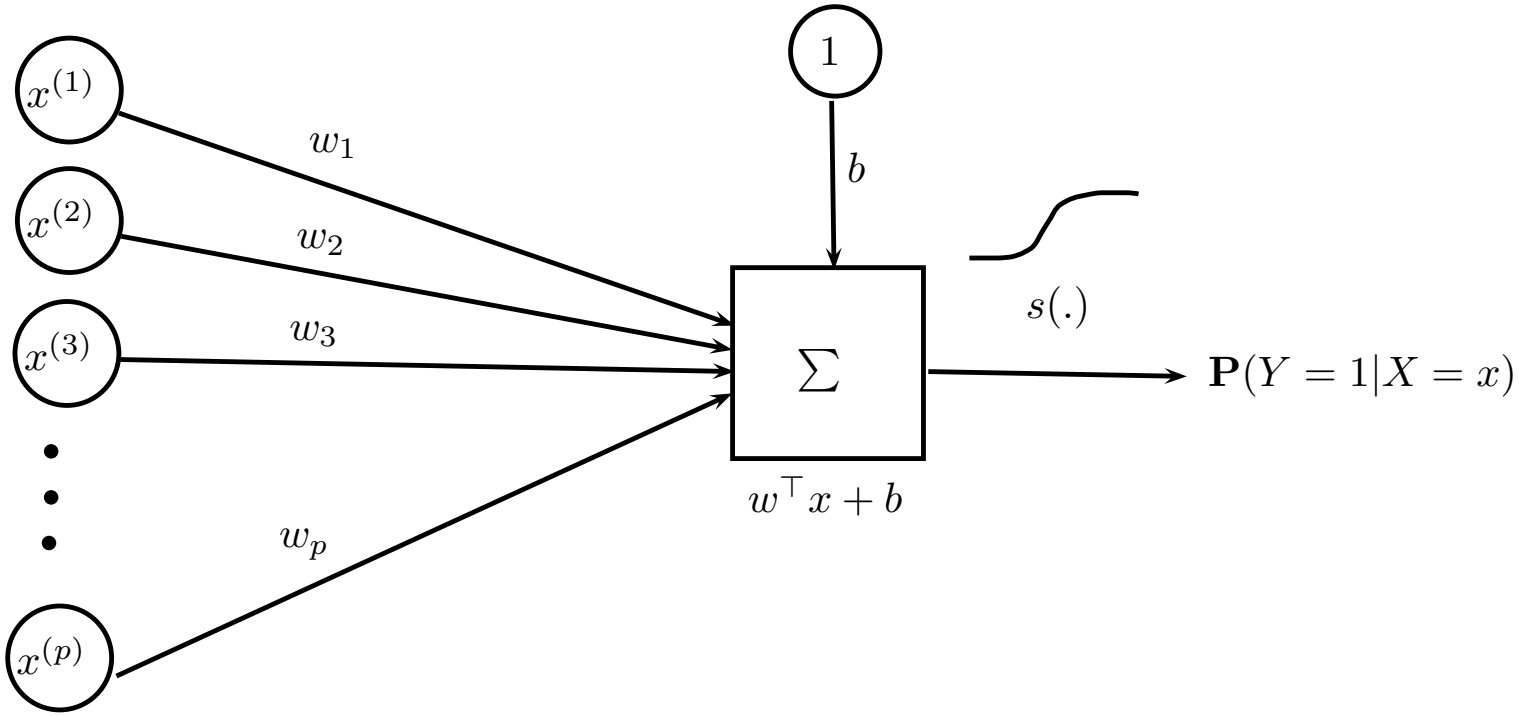
Artificial Neural Networks

Biological inspiration

- Basic computational elements: neurons.
- Receives signals from other neurons via dendrites.
- Sends processed signals via axons.
- Axon-dendrite interactions at synapses.
- $10^{10} - 10^{11}$ neurons.
- $10^{14} - 10^{15}$ synapses.

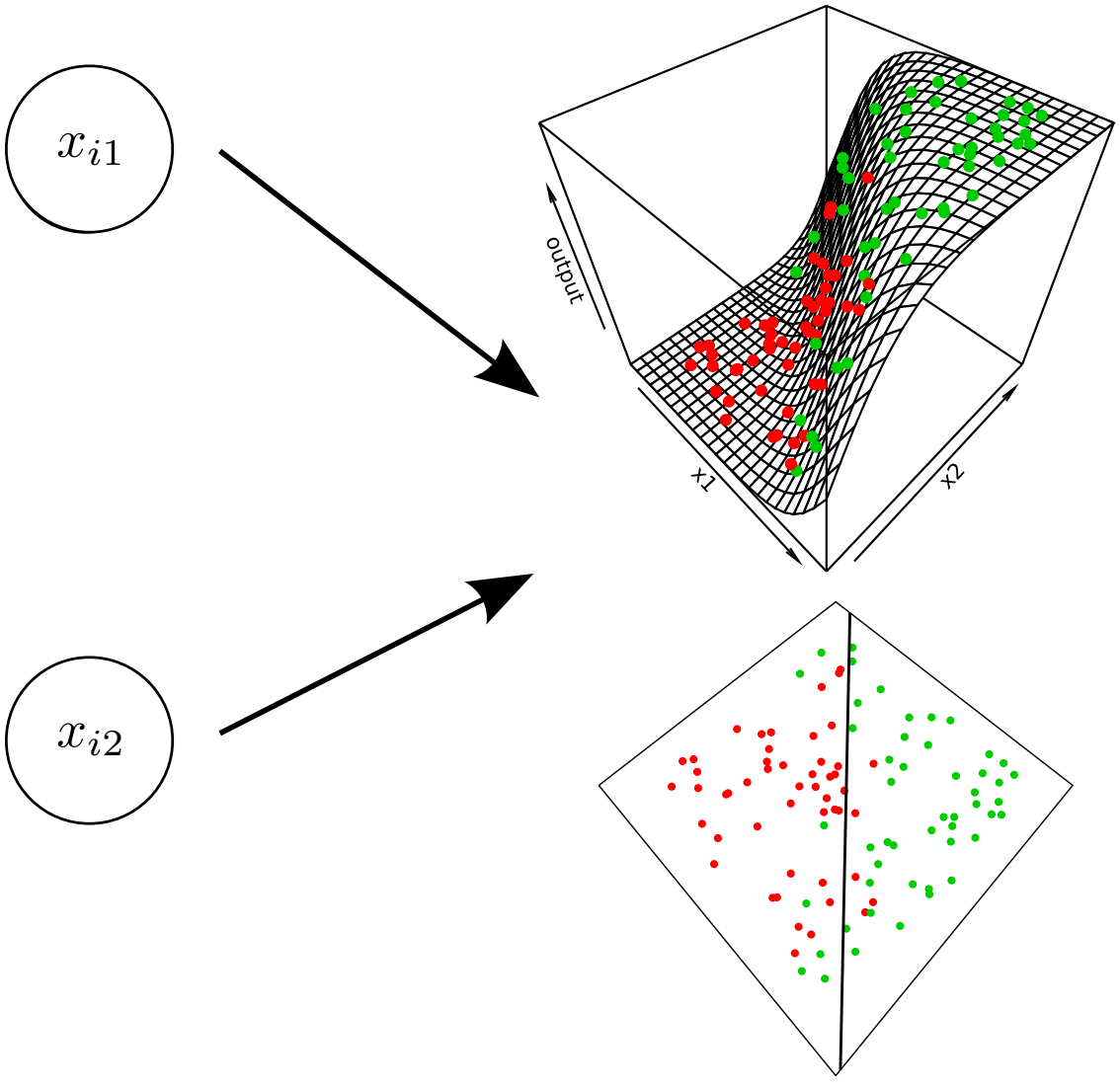


Single Neuron Classifier

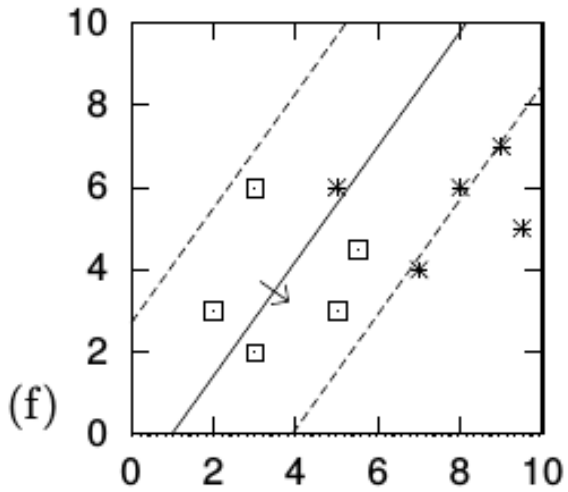


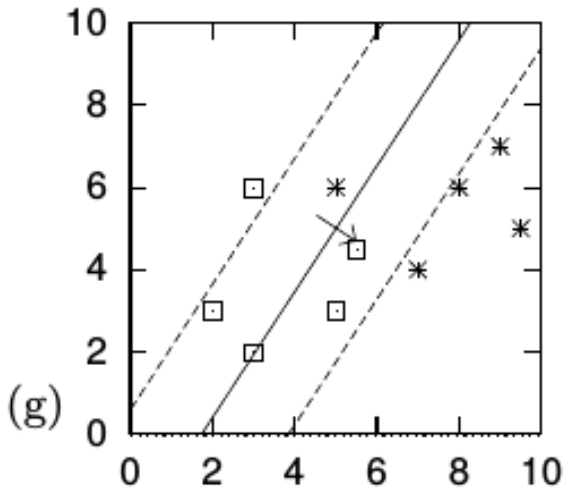
- **activation** $w^\top x + b$ (linear in **inputs** x)
- **activation/transfer function** s gives the **output/activity** (potentially nonlinear in x)
- common nonlinear activation function $s(a) = \frac{1}{1+e^{-a}}$: **logistic regression**
- learn w and b via gradient descent

Single Neuron Classifier

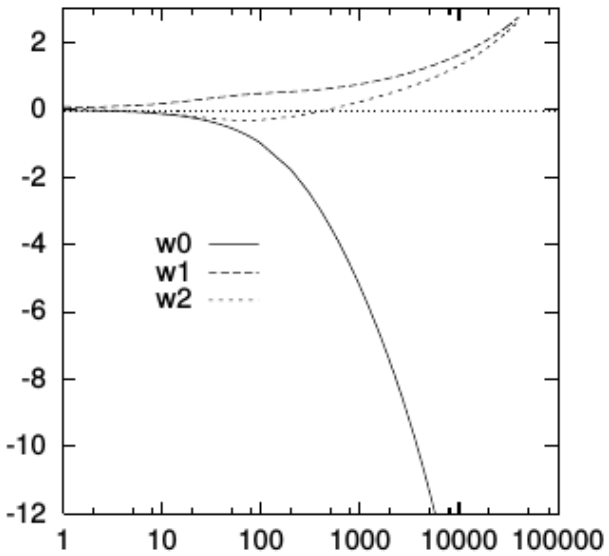


Overfitting



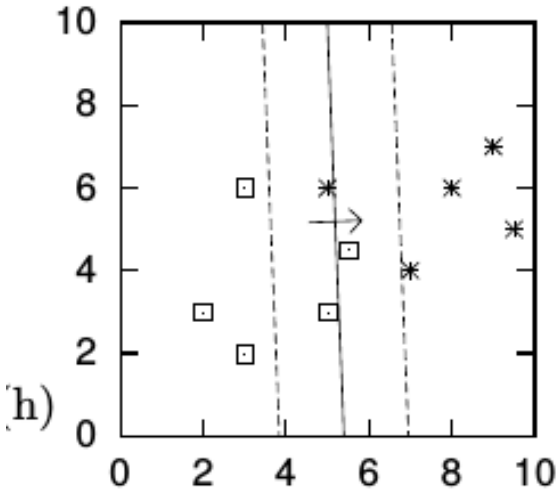


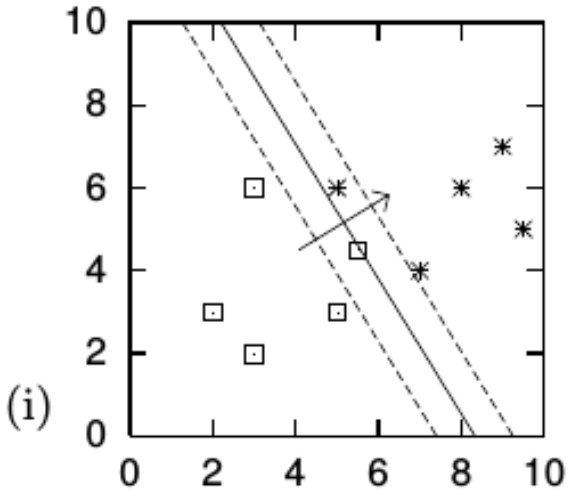
iterations 30,80



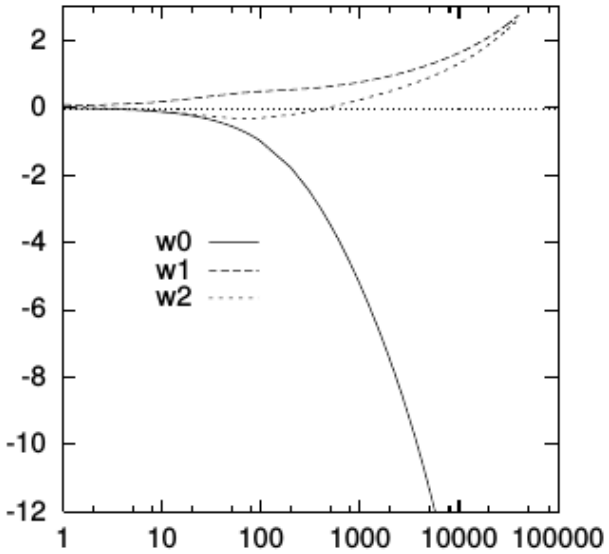
Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

Overfitting



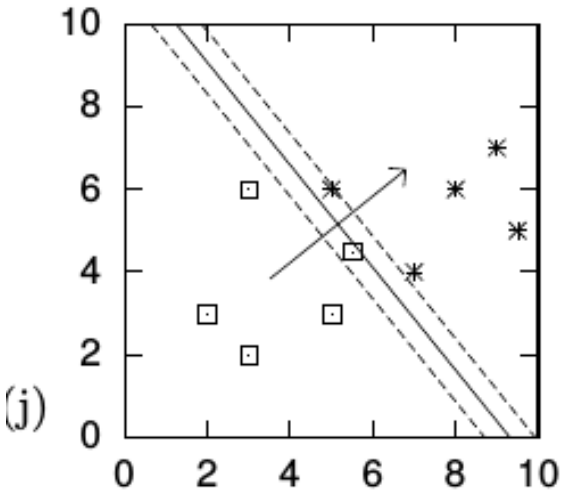


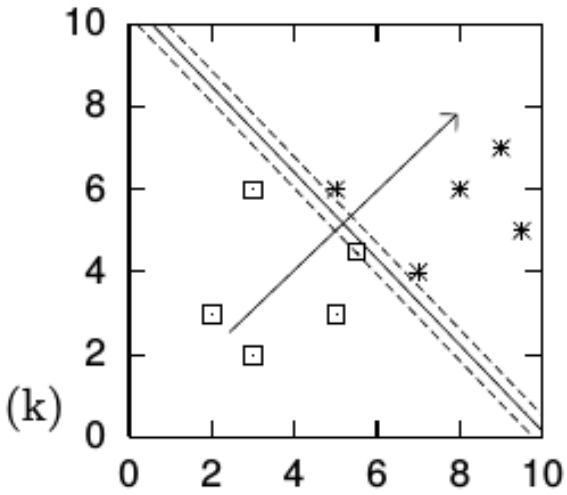
iterations 500,3000

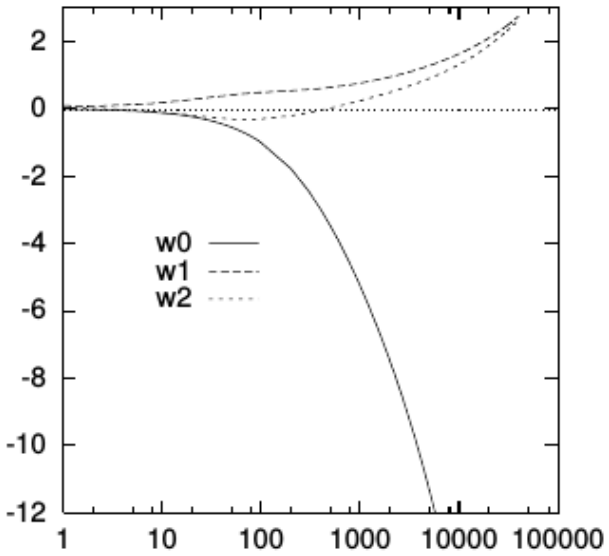


Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

Overfitting







iterations 10000,40000

prevent overfitting by:

- **early stopping**: just halt the gradient descent
- regularization: L_2-regularization called **weight decay** in neural networks literature.

Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

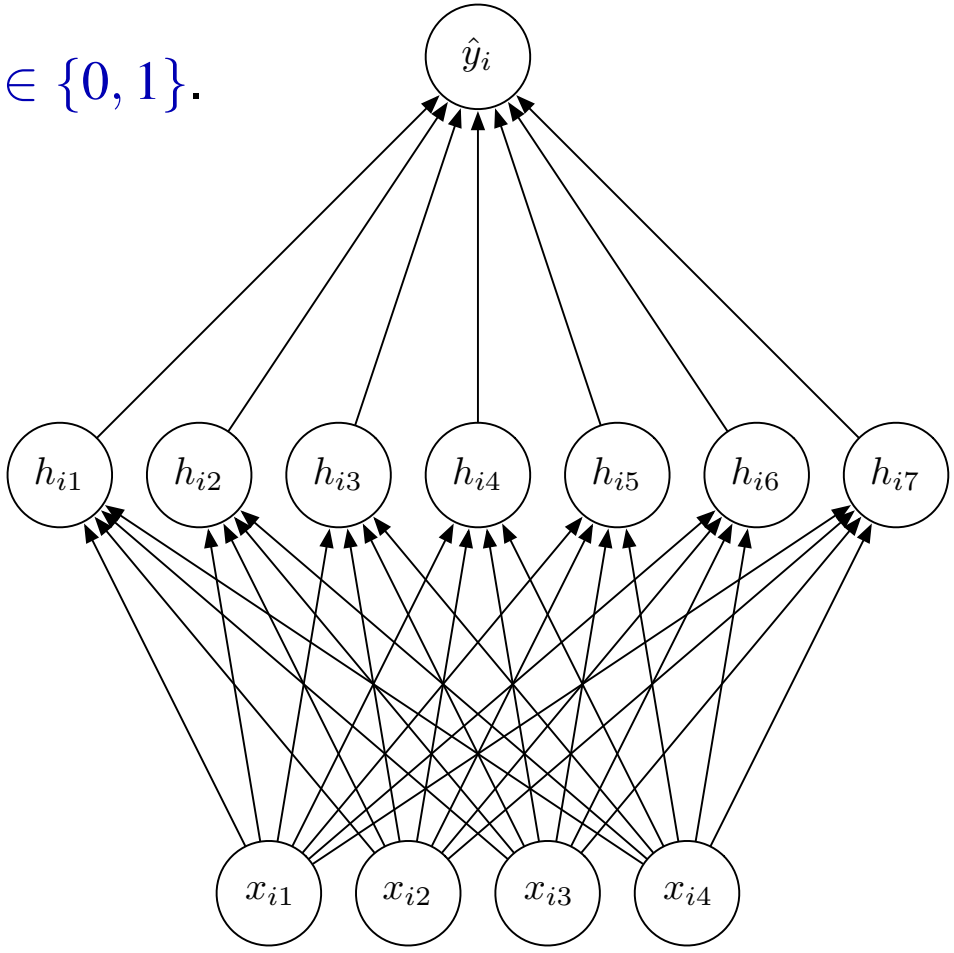
Multilayer Networks

- Data vectors $x_i \in \mathbb{R}^p$, binary labels $y_i \in \{0, 1\}$.
- **inputs** $x_{i1}, \ldots, x_{ip}$
- **output** $\hat{y}_i = \mathbb{P}(Y = 1 | X = x_i)$
- **hidden unit activities** $h_{i1}, \ldots, h_{im}$
  - Compute **hidden unit activities**:

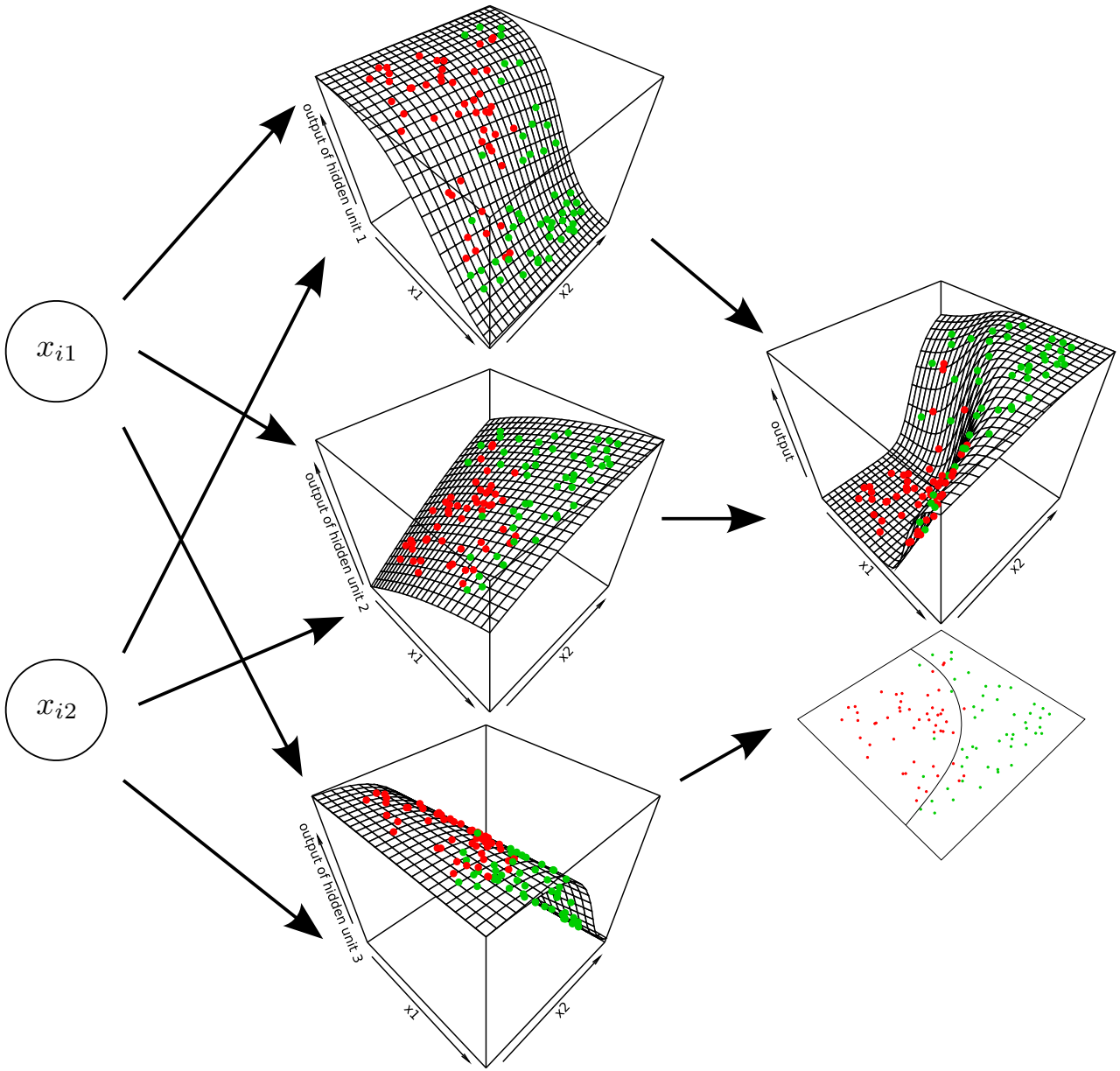
$$h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij}\right)$$

  - Compute **output probability**:

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_k^o h_{il}\right)$$



Multilayer Networks



Training a Neural Network

- Objective function: L_2-regularized log-loss

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) + \frac{\lambda}{2} \left(\sum_{jl} (w_{jl}^h)^2 + \sum_{l} (w_l^o)^2 \right)$$

where

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_l^o h_{il} \right) \qquad h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij} \right)$$

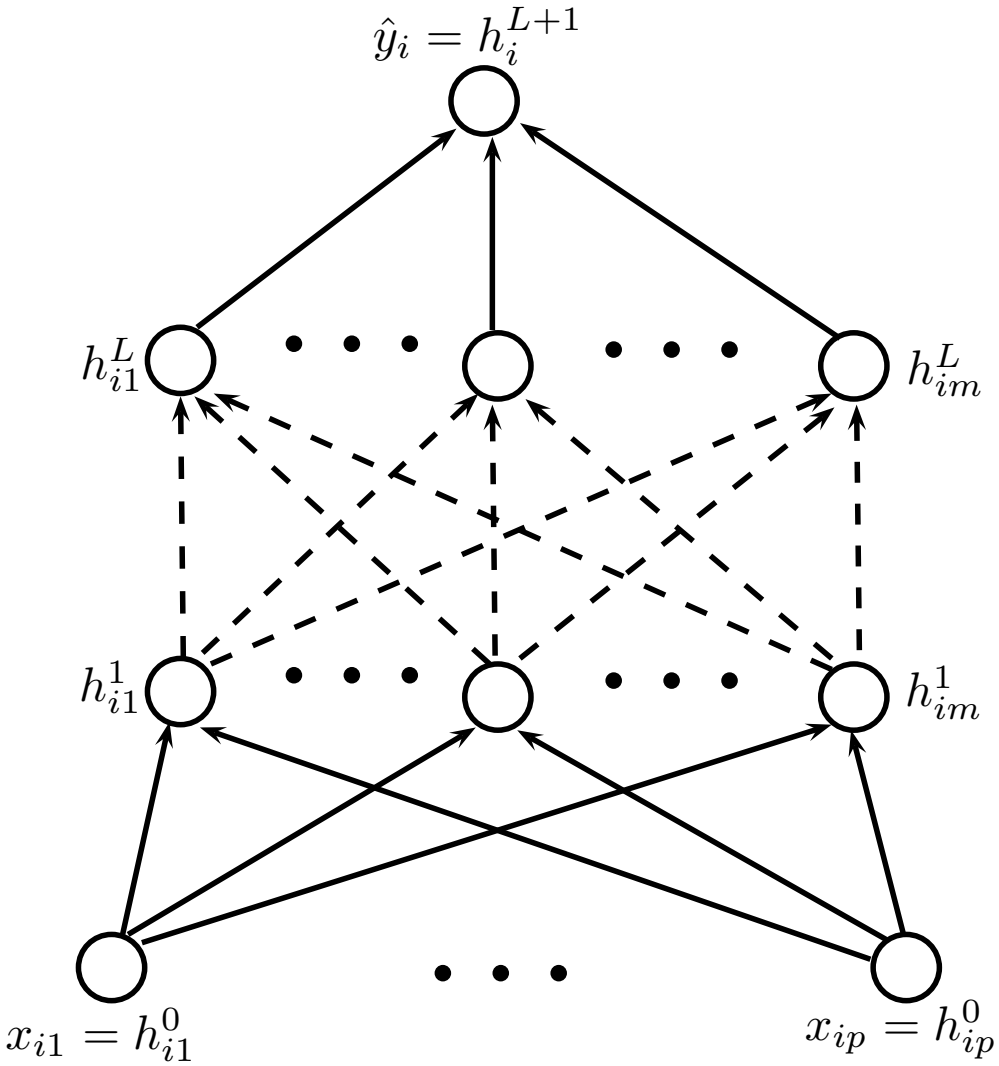
- Optimize parameters $\theta = \left\{ b^h, w^h, b^o, w^o \right\}$, where $b^h \in \mathbb{R}^m$, $w^h \in \mathbb{R}^{p \times m}$, $b^o \in \mathbb{R}$, $w^o \in \mathbb{R}^m$ with gradient descent.

$$\frac{\partial J}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} (\hat{y}_i - y_i) h_{il},$$

$$\frac{\partial J}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial h_{il}} \frac{\partial h_{il}}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} (\hat{y}_i - y_i) w_l^o h_{il} (1 - h_{il}) x_{ij}.$$

- L_2-regularization often called **weight decay**.
- Multiple hidden layers: **Backpropagation** algorithm

Multiple hidden layers

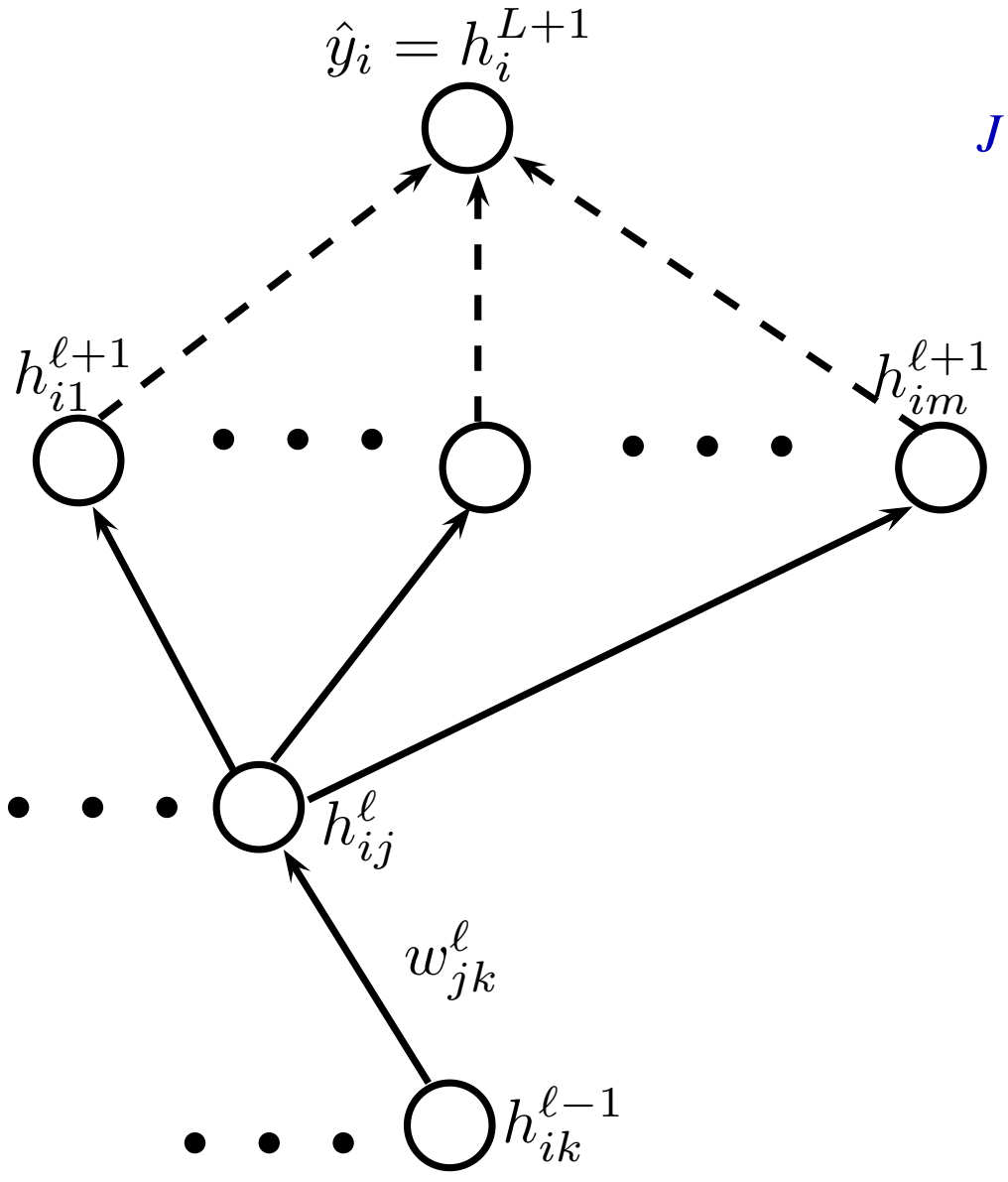


$$h_i^{\ell+1} = \underline{s}\left(W^{\ell+1} h_i^{\ell}\right)$$

- $W^{\ell+1} = \left(w_{jk}^{\ell}\right)_{jk}$: weight matrix at the $(\ell+1)$-th layer, weight w_{jk}^{ℓ} on the edge between $h_{ik}^{\ell-1}$ and h_{ij}^{ℓ}
- $\underline{s}$: entrywise (logistic) transfer function

$$\hat{y}_i = \underline{s}\left(W^{L+1}\underline{s}\left(W^{L}\left(\cdots \underline{s}\left(W^{1} x_i\right)\right)\right)\right)$$

Backpropagation



$$J = -\sum_{i=1}^{n} y_i \log h_i^{L+1} + (1-y_i) \log(1-h_i^{L+1})$$

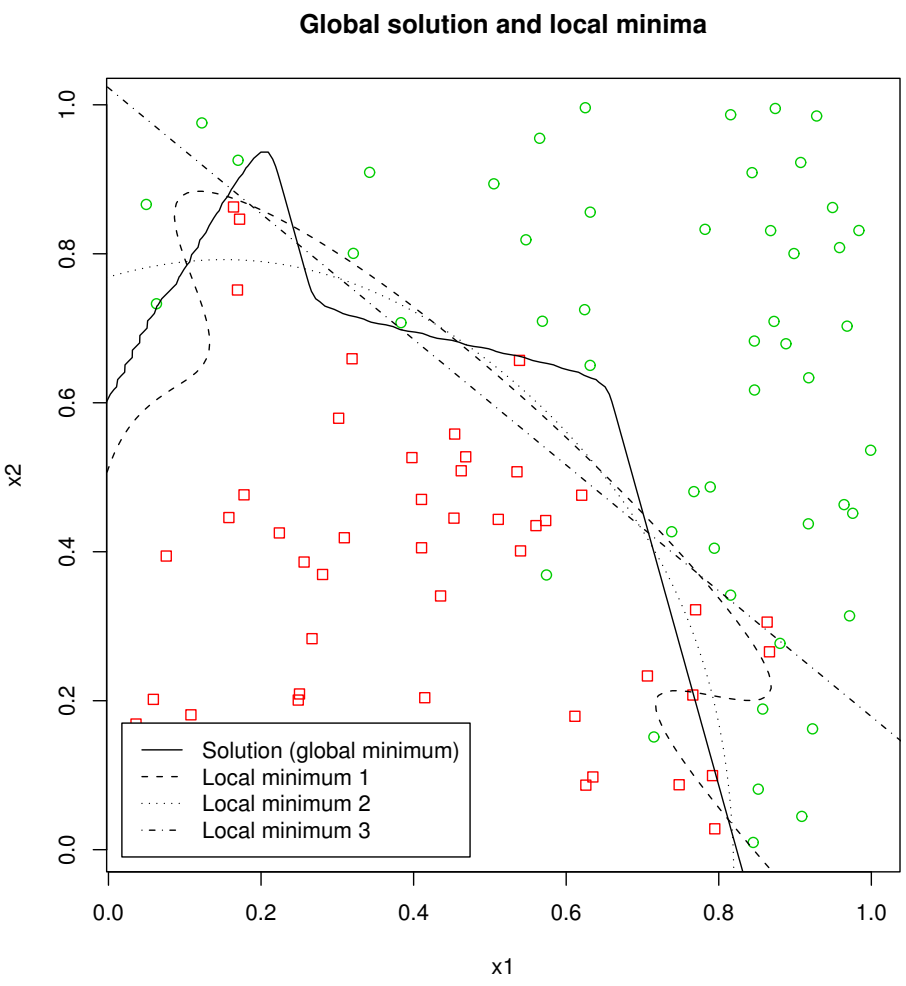
- Gradients wrt h_{ij}^{ℓ} computed by recursive applications of chain rule, and propagated through the network backwards.

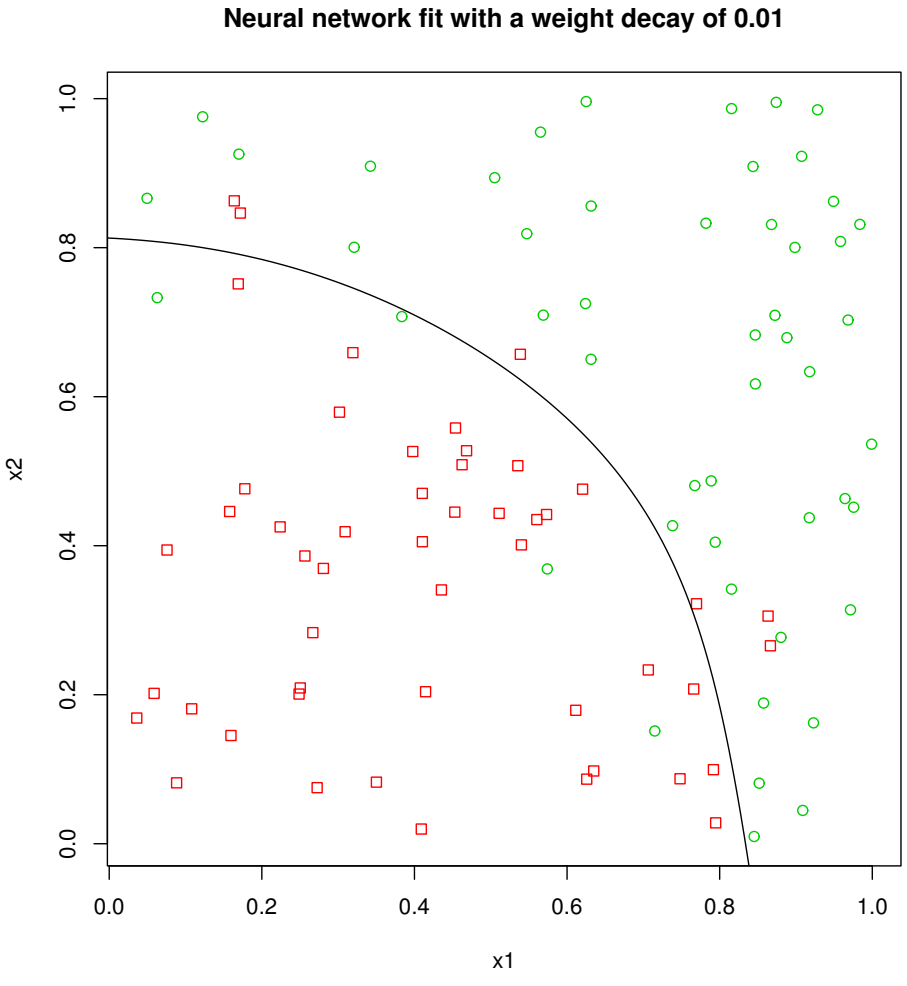
$$\frac{\partial J}{\partial h_i^{L+1}} = -\frac{y_i}{h_i^{L+1}} + \frac{1-y_i}{1-h_i^{L+1}}$$

$$\frac{\partial J}{\partial h_{ij}^{\ell}} = \sum_{r=1}^{m} \frac{\partial J}{\partial h_{ir}^{\ell+1}} \frac{\partial h_{ir}^{\ell+1}}{\partial h_{ij}^{\ell}}$$

$$\frac{\partial J}{\partial w_{jk}^{\ell}} = \sum_{i=1}^{n} \frac{\partial J}{\partial h_{ij}^{\ell}} \frac{\partial h_{ij}^{\ell}}{\partial w_{jk}^{\ell}}$$

Neural Networks





R package implementing neural networks with a single hidden layer: `nnet`.

Neural Networks – Discussion

- Nonlinear hidden units introduce modelling flexibility.
- In contrast to user-introduced nonlinearities, features are global, and can be learned to maximize predictive performance.
- Neural networks with a single hidden layer and sufficiently many hidden units can model arbitrarily complex functions.
- Optimization problem is **not convex**, and objective function can have many local optima, plateaus and ridges.
- On large scale problems, often use **stochastic gradient descent**, along with a whole host of techniques for optimization, regularization, and initialization.
- Recent developments, especially by Geoffrey Hinton, Yann LeCun, Yoshua Bengio, Andrew Ng and others. See also `http://deeplearning.net/`.

Dropout Training of Neural Networks

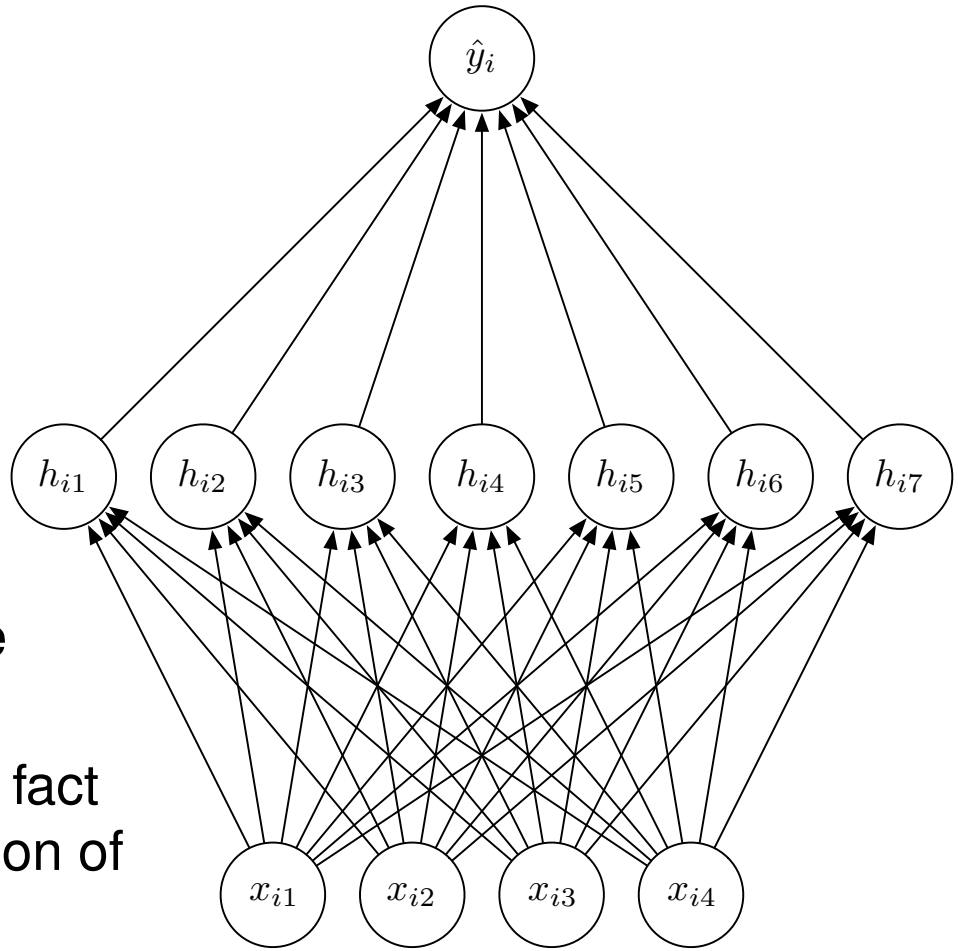
- Neural network with single layer of hidden units:
  - **Hidden unit activations**:

$$h_{ik} = s\left(b_k^h + \sum_{j=1}^{p} W_{jk}^h x_{ij}\right)$$

  - **Output probability**:

$$\hat{y}_i = s\left(b^o + \sum_{k=1}^{m} W_k^o h_{ik}\right)$$

- Large, overfitted networks often have co-adapted hidden units.
- What each hidden unit learns may in fact be useless, e.g. predicting the negation of predictions from other units.
- Can prevent co-adaptation by randomly **dropping out** units from network.

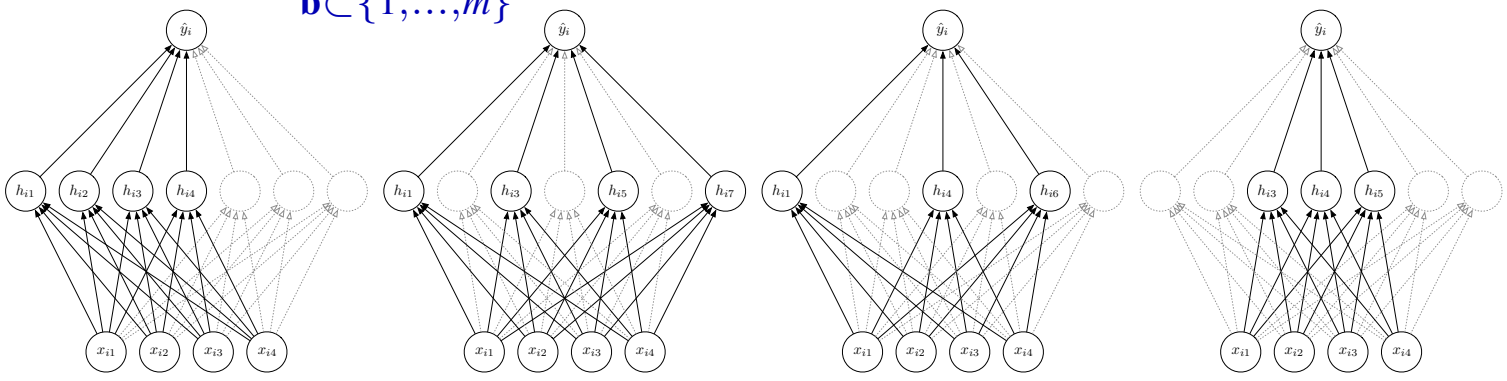


Hinton et al (2012).

Dropout Training of Neural Networks

- Model as an ensemble of networks:

$$p(y_i = 1|x_i, \theta) = \sum_{\mathbf{b} \subset \{1,\ldots,m\}} q^{|\mathbf{b}|}(1-q)^{m-|\mathbf{b}|} p(y_i = 1|x_i, \theta, \text{drop out units } \mathbf{b})$$



- **Weight-sharing** among all networks: each network uses a subset of the parameters of the full network (corresponding to the retained units).
- Training by stochastic gradient descent: at each iteration a network is sampled from ensemble, and its subset of parameters are updated.
- Biological inspiration: 10^{14} weights to be fitted in a lifetime of 10^9 seconds
  - Poisson spikes as a regularization mechanism which prevents co-adaptation: Geoff Hinton on Brains, Sex and Machine Learning

Dropout Training of Neural Networks

Classification of phonemes in speech.

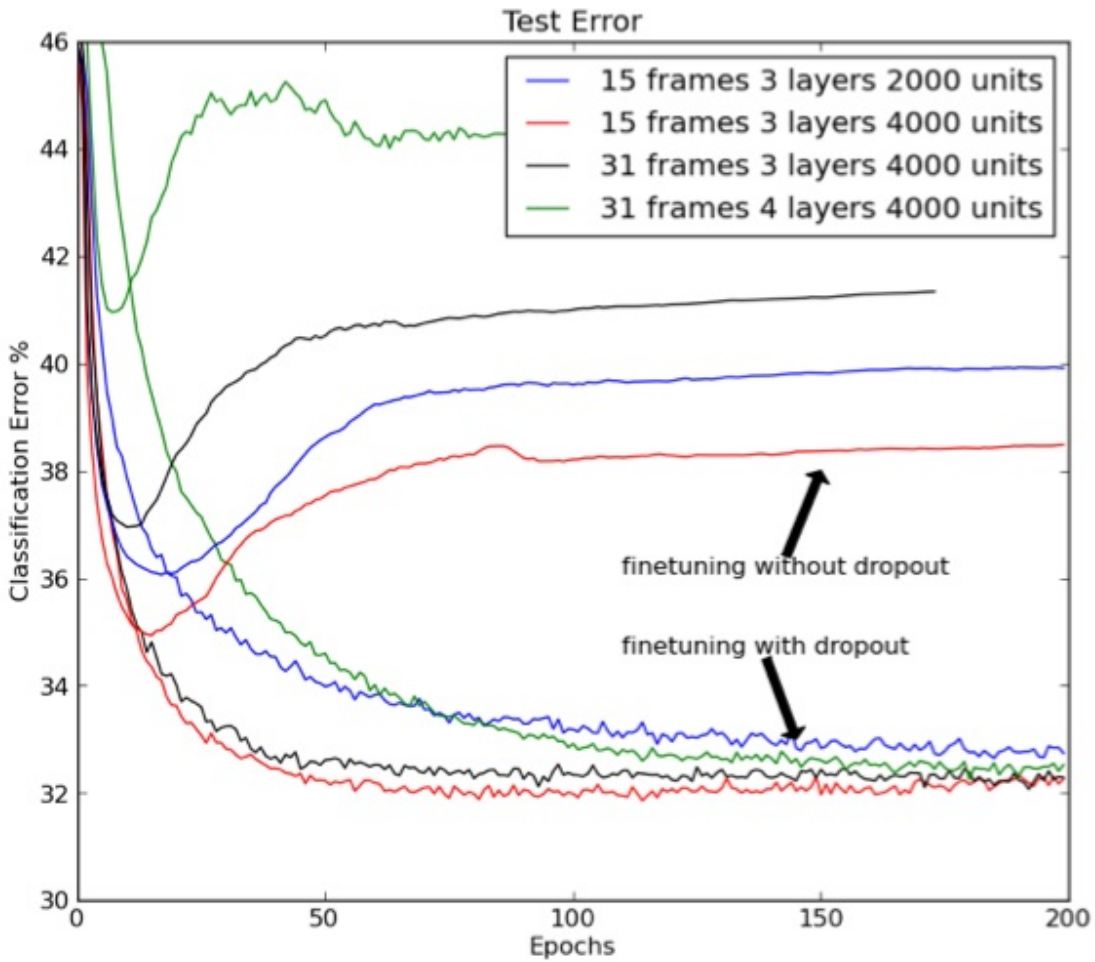


Figure from Hinton et al.