EE 7207 – Neural & Fuzzy Systems

Dr Poh Eng Kee

Professor (Adjunct)

e-mail: eekpoh@ntu.edu.sg

S2.1-B2-17

Common Faculty Office

Machine Learning Algorithms

Machine Learning Algorithms

I. Supervised Learning

II. Unsupervised Learning

III. Others : Reinforcement Learning

LECTURE OUTLINE

Supervised Learning

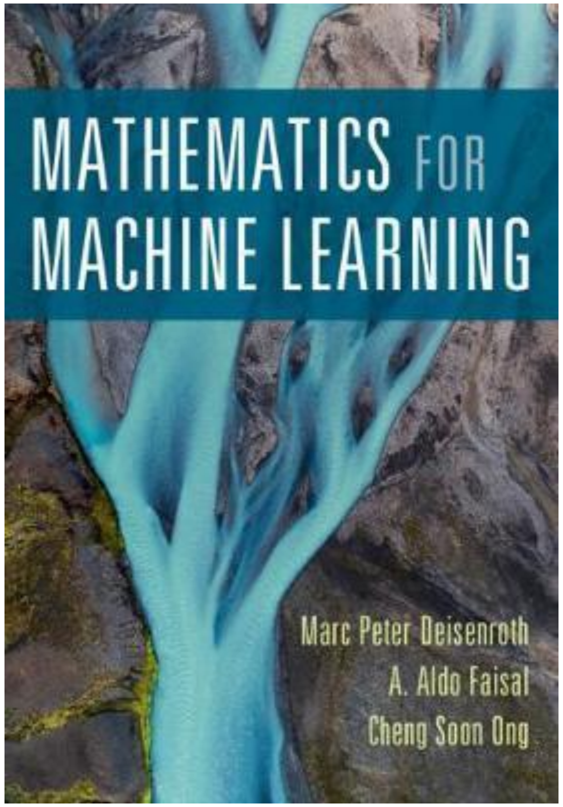
1. *Linear Regression : Linear Least Squares*
2. Classification :
   - Linear Discriminant Analysis
   - Logistics Regression
   - *Neural Network : Backpropagation using Gradient Descent, Stochastic Descent*

Unsupervised Learning

3. Classification :
   - *Support Vector Machines : Constrained Optimization*
4. Clustering - K-Means
5. *Dimension Reduction : Principal Component Analysis*

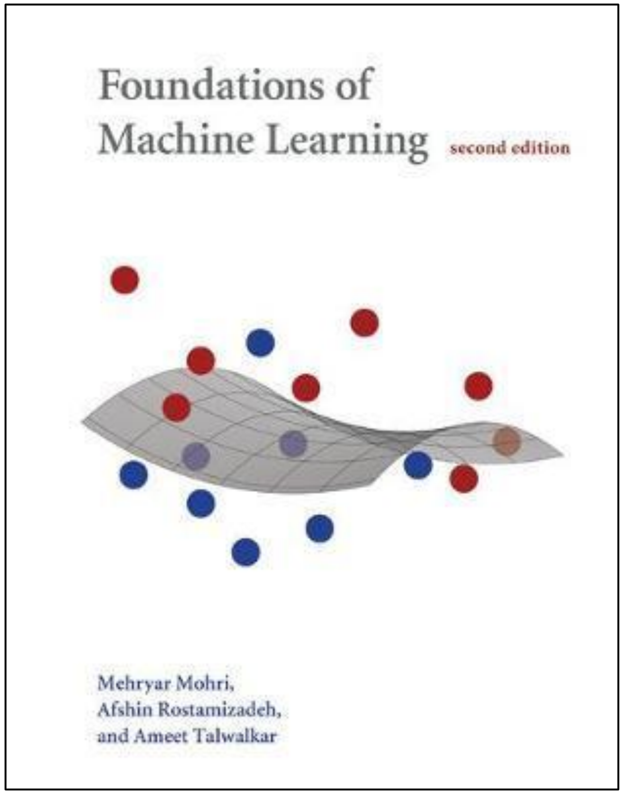
* Only highlighted in red is covered in lecture. Stochastic/Bayesian perspective is not included.

Reference Text



The fundamental mathematical tools needed to understand machine learning include linear algebra, analytic geometry, matrix decompositions, vector calculus, optimization, probability and statistics. This self-contained textbook bridges the gap between mathematical and machine learning texts, introducing the mathematical concepts with a minimum of prerequisites. It uses these concepts to derive four central machine learning methods: linear regression, principal component analysis, Gaussian mixture models and support vector machines. For students and others with a mathematical background, these derivations provide a starting point to machine learning texts. For those learning the mathematics for the first time, the methods help build intuition and practical experience with applying mathematical concepts. Every chapter includes worked examples and exercises to test understanding. Programming tutorials are offered on the book's web site.

Reference Text



This book is a general introduction to machine learning that can serve as a textbook for graduate students and a reference for researchers. It covers fundamental modern topics in machine learning while providing the theoretical basis and conceptual tools needed for the discussion and justification of algorithms. It also describes several key aspects of the application of these algorithms. The authors aim to present novel theoretical tools and concepts while giving concise proofs even for relatively advanced topics.

Foundations of Machine Learning is unique in its focus on the analysis and theory of algorithms. The first four chapters lay the theoretical foundation for what follows; subsequent chapters are mostly self-contained. Topics covered include the Probably Approximately Correct (PAC) learning framework; generalization bounds based on Rademacher complexity and VC-dimension; Support Vector Machines (SVMs); kernel methods; boosting; on-line learning; multi-class classification; ranking; regression; algorithmic stability; dimensionality reduction; learning automata and languages; and reinforcement learning.

I. **SUPERVISED LEARNING**

Supervised Learning

- In supervised learning, we are given a data set and already know what our correct output should look like (**label**), having the idea that there is a relationship between the input and the output.
- Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

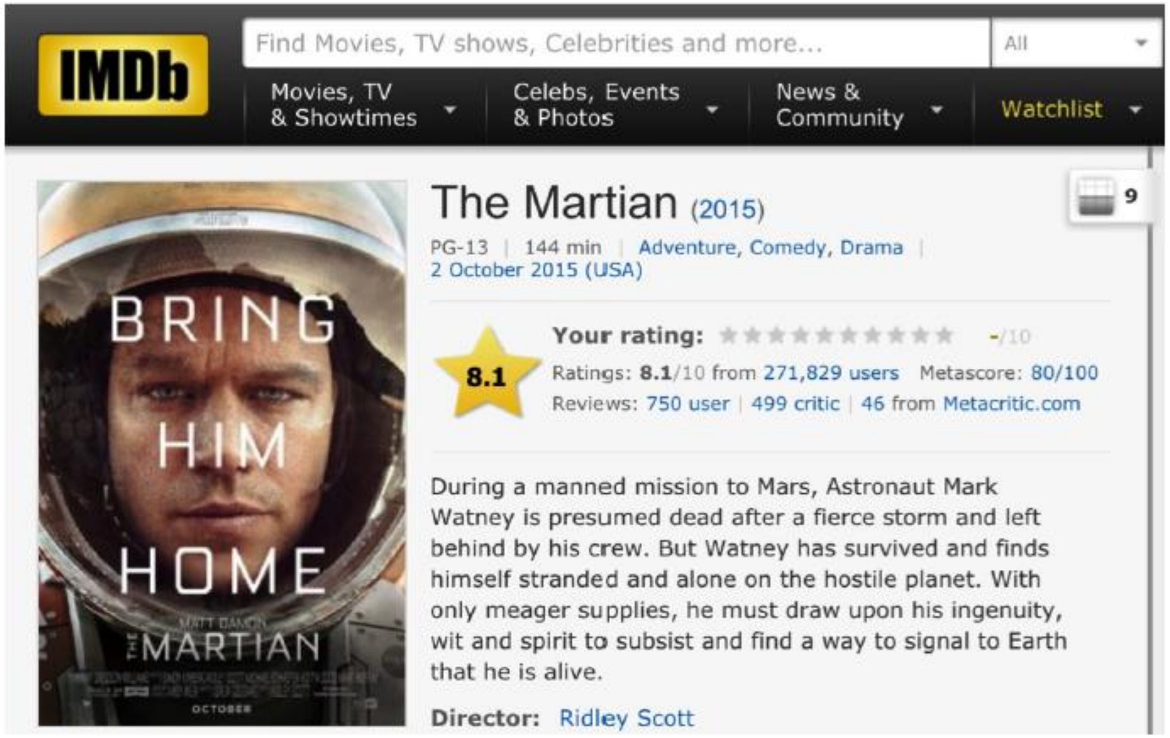
Supervised Learning

- **Example :**
- Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a **Regression** problem.
- We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories. Solve using **Neural Network** using **Backpropagation** algorithm.

1. Linear Regression : Linear Least Squares

Applications of Regression

- What should I watch this Friday?
- **Goal**: Predict movie rating automatically!
- **Goal:** How many followers will I get?
- **Goal:** Predict the price of the house



The Martian (2015)

PG-13 | 144 min | Adventure, Comedy, Drama |
2 October 2015 (USA)

8.1 Your rating: ★★★★★★★★★★ -/10

Ratings: 8.1/10 from 271,829 users Metascore: 80/100

Reviews: 750 user | 499 critic | 46 from Metacritic.com

During a manned mission to Mars, Astronaut Mark Watney is presumed dead after a fierce storm and left behind by his crew. But Watney has survived and finds himself stranded and alone on the hostile planet. With only meager supplies, he must draw upon his ingenuity, wit and spirit to subsist and find a way to signal to Earth that he is alive.

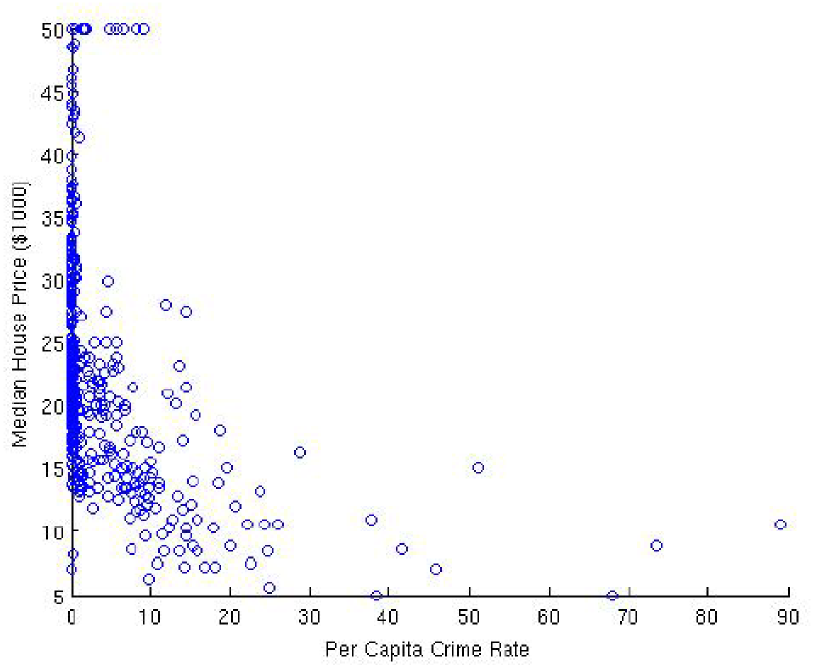
Director: Ridley Scott

- What do all these problems have in common?
  - Continuous outputs, we'll call these *y*

    (e.g., rating: a real number between 0-10,# of followers, house price)
- Predicting continuous outputs is called regression
- How to predict these outputs?
  - Features (inputs), we denotes these as t (or $\boldsymbol{t}$ if vectors)
  - Training data, many t_i for which y_i is known (e.g. the movies for which we know the rating)
  - A model, a function that represents the relationship between t and y.
  - A loss or a cost or objective function, which tells us how well our model approximates the training data
  - Optimization, a mathematical method of finding the parameters of our model that minimizes the loss function

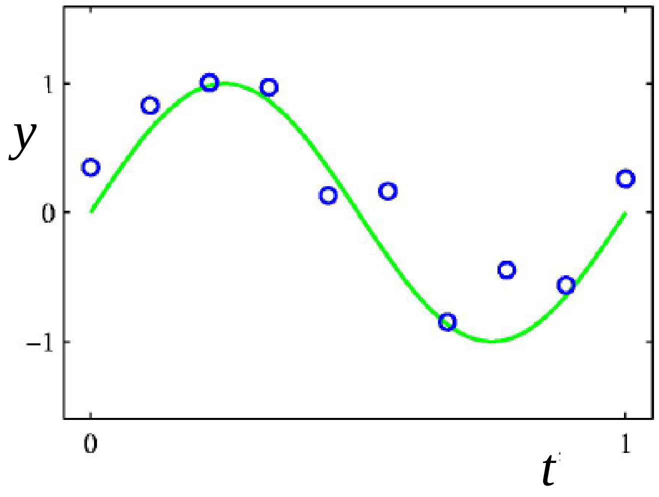
E.g. Boston Housing Data – Good Feature

- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



- Use this to predict house prices in other neighborhoods
- Is this a good input (attribute) to predict house prices?

1-D Regression



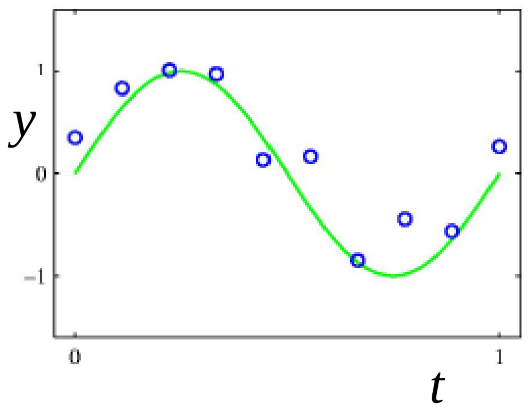
- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in x, but may be displaced in y

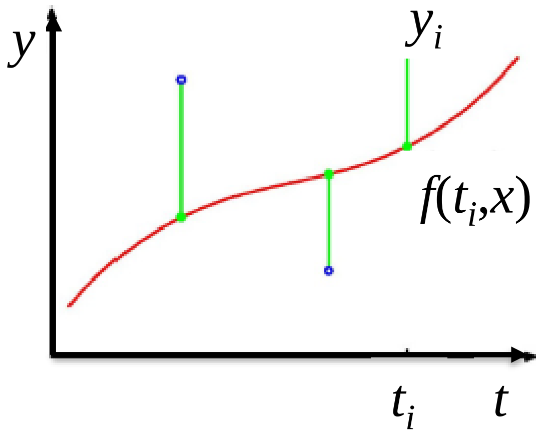
$$y = f(t) + \varepsilon$$

with ε some noise

- In green is the "true" curve that we don't know
- Goal: We want to fit a curve to these points

Linear Regression with Single Input (Univariate Linear Regression)





- Define a model

$$\text{Polynomial Model}: f(t,x) = x_1 + x_2 t + \ldots + x_n t^{n-1}$$

- Standard loss/cost/objective function measures the squared error between f and the true value y

$$\text{Loss Function}: l(x) = \sum_{i=1}^{m} [y_i - (x_1 + x_2 t_i + \ldots + x_n t_i^{n-1})]^2$$

- The loss for the red hypothesis is the **sum of the squared vertical errors** (squared lengths of green vertical lines)

Linear Least Squares to determine unknown parameter x

In general, let $b = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, A = \begin{bmatrix} 1 & t_1 & \dots & t_1^{n-1} \\ 1 & t_2 & \dots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & \dots & t_m^{n-1} \end{bmatrix}$

Better written $\boldsymbol{Ax} \cong \boldsymbol{b}$, since equality usually not exactly satisfiable when $m > n$

Least squares solution $\boldsymbol{x}$ minimizes squared Euclidean norm of residual vector $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax}$,

$$\min_{\boldsymbol{x}} \|\boldsymbol{r}\|_2^2 = \min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{Ax}\|_2^2$$

Existence and Uniqueness

Linear least squares problem $\boldsymbol{A}\boldsymbol{x} \cong \boldsymbol{b}$ *always* has solution

Solution unique if, and only if, columns of $\boldsymbol{A}$ linearly independent, i.e., rank$(\boldsymbol{A}) = n$, where $\boldsymbol{A}$ is $m \times n$

If rank$(\boldsymbol{A}) < n$, then $\boldsymbol{A}$ is *rank-deficient*, and solution of linear least squares problem is not unique

For now, assume $\boldsymbol{A}$ has full column rank n

Normal Equations

Least squares minimizes squared Euclidean norm

$$\|\boldsymbol{r}\|_2^2 = \boldsymbol{r}^T\boldsymbol{r}$$

of residual vector

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$$

To minimize

$$\begin{aligned}\|\boldsymbol{r}\|_2^2 &= \boldsymbol{r}^T\boldsymbol{r} = (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \\ &= \boldsymbol{b}^T\boldsymbol{b} - 2\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{b} + \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x},\end{aligned}$$

take derivative with respect to $\boldsymbol{x}$ and set to $\boldsymbol{o}$,

$$2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{b} = \boldsymbol{o},$$

which reduces to $n \times n$ linear system

Inverse exists if rank(A)=n

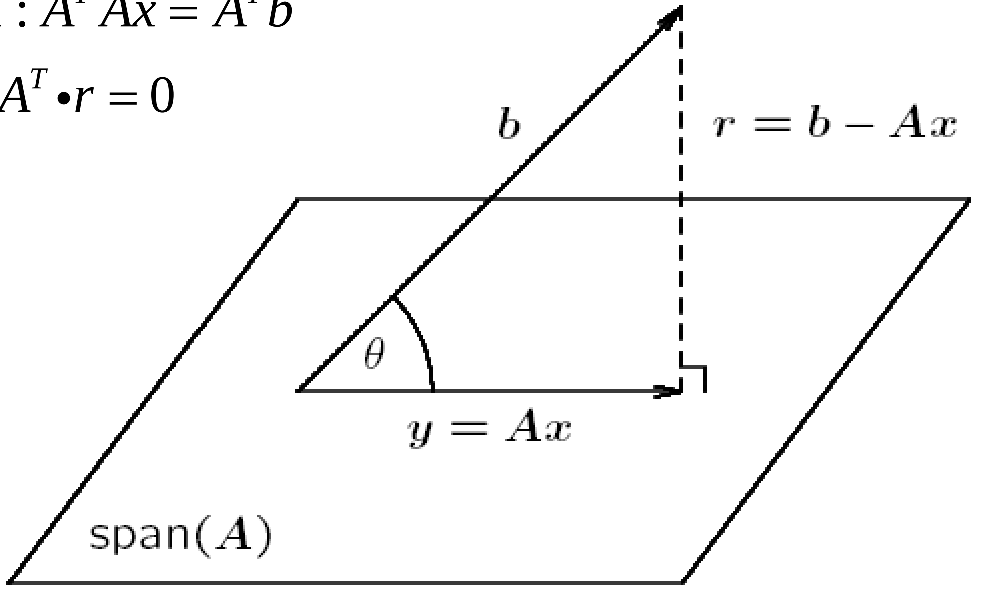
$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \qquad \Rightarrow x = \left[A^T A\right]^{-1} A^T b$$

known as system of *normal equations*

Orthogonality Principle

Normal Equation : $A^T Ax = A^T b$

$\Rightarrow A^T \left(b - Ax\right) = A^T \bullet r = 0$



Example: Data Fitting

Given m data points (t_i, y_i), find n-vector $\boldsymbol{x}$ of parameters that gives "best fit" to model function $f(t, \boldsymbol{x})$:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{m} (y_i - f(t_i, \boldsymbol{x}))^2$$

Problem *linear* if function f linear in components of $\boldsymbol{x}$:

$$f(t, \boldsymbol{x}) = x_1\phi_1(t) + x_2\phi_2(t) + \cdots + x_n\phi_n(t),$$

where functions ϕ_j depend only on t

Written in matrix form as $\boldsymbol{A}\boldsymbol{x} \cong \boldsymbol{b}$, with $a_{ij} = \phi_j(t_i)$ and $b_i = y_i$

Example: Data Fitting

Polynomial fitting,

$$f(t, \boldsymbol{x}) = x_1 + x_2 t + x_3 t^2 + \cdots + x_n t^{n-1},$$

is linear, since polynomial linear in coefficients, though nonlinear in independent variable t

Fitting sum of exponentials, with

$$f(t, \boldsymbol{x}) = x_1 e^{x_2 t} + \cdots + x_{n-1} e^{x_n t},$$

is nonlinear problem

For now, consider only linear least squares problems

Example: Data Fitting

Fitting quadratic polynomial to five data points gives linear least squares problem

$$\boldsymbol{Ax} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \\ 1 & t_5 & t_5^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cong \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \boldsymbol{b}$$

Matrix with columns (or rows) successive powers of independent variable called *Vandermonde* matrix

Example: Data Fitting

For data

t	-1.0	-0.5	0.0	0.5	1.0
y	1.0	0.5	0.0	0.5	2.0

overdetermined 5×3 linear system is

$$Ax = \begin{bmatrix} 1 & -1.0 & 1.0 \\ 1 & -0.5 & 0.25 \\ 1 & 0.0 & 0.0 \\ 1 & 0.5 & 0.25 \\ 1 & 1.0 & 1.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cong \begin{bmatrix} 1.0 \\ 0.5 \\ 0.0 \\ 0.5 \\ 2.0 \end{bmatrix} = b$$

Using least squares solution, we obtain

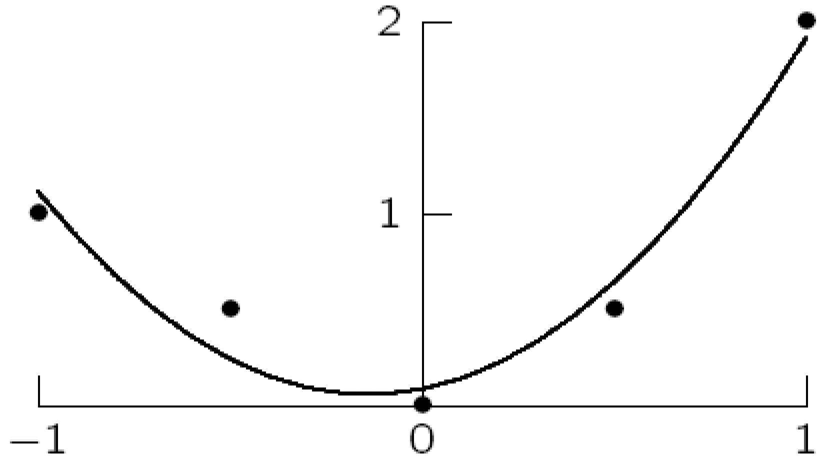
$$x = [0.086 \quad 0.40 \quad 1.4]^T,$$

so approximating polynomial is

$$p(t) = 0.086 + 0.4t + 1.4t^2$$

Example: Data Fitting

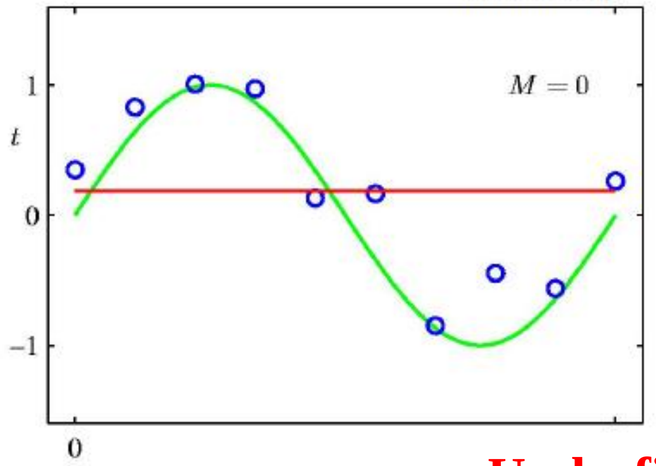
Resulting curve and original data points shown in graph:

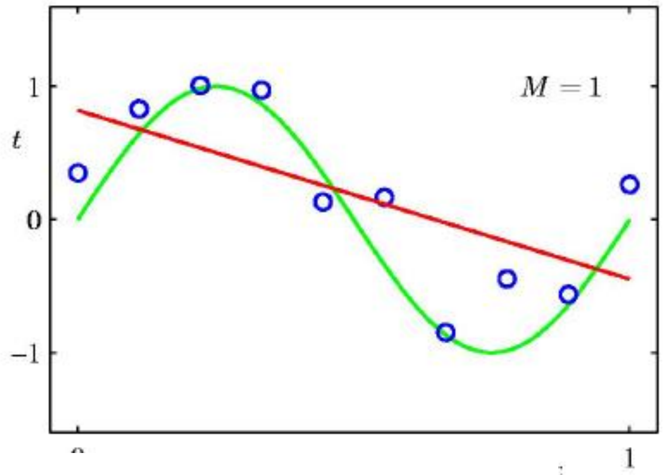


Which Fit is Best?

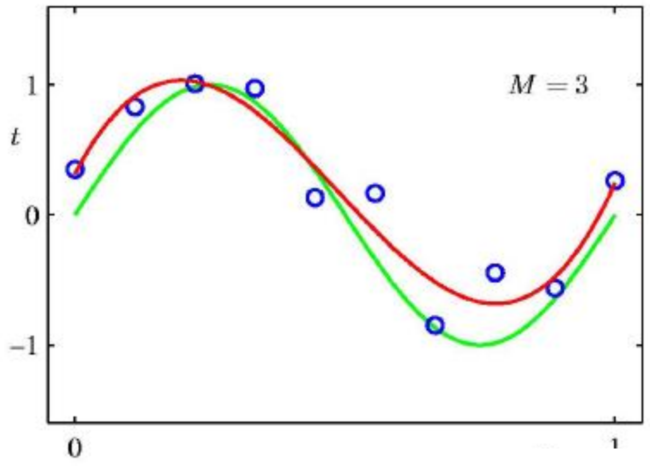
Polynomial Model : $f\left(t,x\right)=x_0+\sum_{i=1}^{M}x_i t^i$

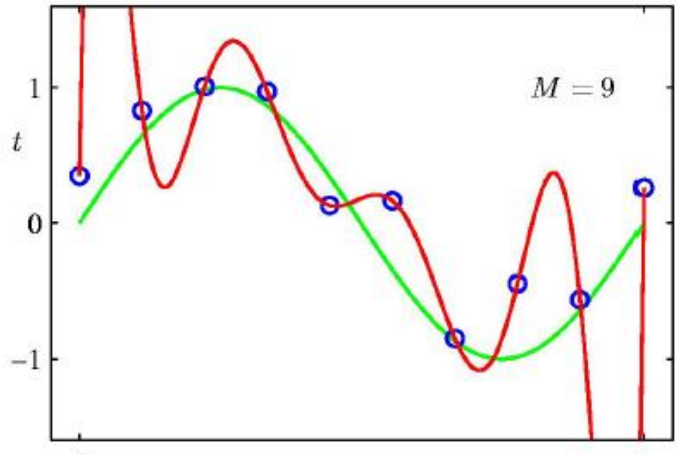
from Bishop





Underfit – Too Simple Models





Overfit – Overly Complex Model

Technical Explanation of Bias-Variance Trade-Off: Mean Square Error (MSE) Estimate

Let $\widehat{\theta}$ denote the random variable of the associated estimator for the unknown parameter θ whose true value is θ_0. The mean-square error (MSE) is defined as (Cramer-Rao Lower bound on left side)

$$\frac{-1}{E\left[\frac{\partial^2}{\partial\theta^2}\ln p(x;\theta)\right]} \le \mathrm{MSE} = \mathrm{E}\left[\left(\widehat{\theta}-\theta_0\right)^2\right]$$

$$= \mathrm{E}\left[\left(\left(\widehat{\theta}-\mathrm{E}\left[\widehat{\theta}\right]\right)+\left(\mathrm{E}\left[\widehat{\theta}\right]-\theta_0\right)\right)^2\right]$$

$$= \underbrace{\mathrm{E}\left[\left(\widehat{\theta}-\mathrm{E}\left[\widehat{\theta}\right]\right)^2\right]}_{\text{Variance}} + \underbrace{\left(\mathrm{E}\left[\widehat{\theta}\right]-\theta_0\right)^2}_{\text{Bias}^2}$$

Bias-Variance Tradeoff

- The MSE comprises 2 terms
  - First one is contributed from the variance of the estimator around its mean value. Second one from the bias due to the difference of the mean value of the estimator from the true one.
  - For a fixed number of training points, N, in the data set X, trying to minimize the variance term results in increase in the bias term and vice versa. This is because to reduce the bias term, one has to increase the complexity (more free parameters) which results in higher variance as we change the training sets due to overfitting.
  - This is known as the bias-variance dilemma/tradeoff in estimation. Usually, we refer to it as Occam's razor rule.
  - Only way to reduce both terms simultaneously is to increase N and increase the complexity of the model judiciously to minimise the overall MSE.
  - Generally, linear models have low variance but high bias (underfitting risk). High variance-low bias models (overfitting risk) such as ***Decision Trees*** are preferred if data is abundant.

2. Neural Network: Backpropagation Using Steepest Descent and Stochastic Gradient

Optimization

Given function $f\colon \mathbb{R}^n \to \mathbb{R}$, and set $S \subseteq \mathbb{R}^n$, find $\boldsymbol{x}^* \in S$ such that $f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$

$\boldsymbol{x}^*$ called *minimizer* or *minimum* of f

Suffices to consider only minimization, since maximum of f is minimum of $-f$

Objective function f usually differentiable, may be linear or nonlinear

Constraint set S defined by system of equations and inequalities that may be linear or nonlinear

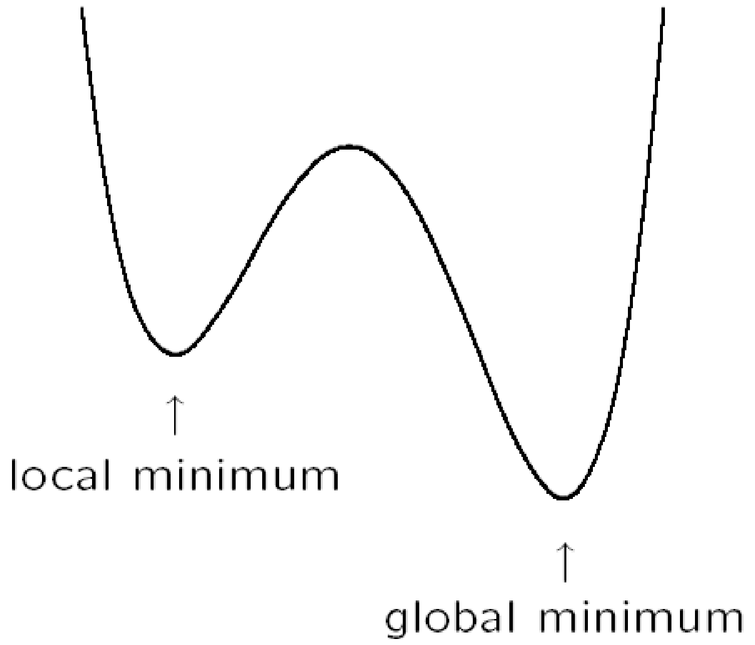
Points $\boldsymbol{x} \in S$ called *feasible* points

If $S = \mathbb{R}^n$, problem is *unconstrained*

Local vs Global Optimization

$\boldsymbol{x}^* \in S$ is *global minimum* if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$

$\boldsymbol{x}^* \in S$ is *local minimum* if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$ in some neighborhood of $\boldsymbol{x}^*$



Global Optimization

Finding, or even verifying, global minimum is difficult, in general

Most optimization methods designed to find local minimum, which may or may not be global minimum

If global minimum desired, can try several widely separated starting points and see if all produce same result

For some problems, such as linear programming, global optimization tractable

Existence of Minimum

If f is continuous on *closed* and *bounded* set $S \subseteq \mathbb{R}^n$, then f has global minimum on S

If S is not closed or is unbounded, then f may have no local or global minimum on S

Continuous function f on unbounded set $S \subseteq \mathbb{R}^n$ is *coercive* if

$$\lim_{\|\boldsymbol{x}\| \to \infty} f(\boldsymbol{x}) = +\infty,$$

i.e., $f(\boldsymbol{x})$ must be large whenever $\|\boldsymbol{x}\|$ is large

If f coercive on closed, unbounded set $S \subseteq \mathbb{R}^n$, then f has global minimum on S

Unconstrained Optimization: 1st Order Necessary Condition for Local Minimum

Corollary 4 *Suppose $f(x)$ is differentiable at $\bar{x}$. If $\bar{x}$ is a local minimum, then $\nabla f(\bar{x}) = 0$.*

Proof: If it were true that $\nabla f(\bar{x}) \neq 0$, then $d = -\nabla f(\bar{x})$ would be a descent direction, whereby $\bar{x}$ would not be a local minimum. ∎

The above corollary is a *first order necessary optimality condition* for an unconstrained minimization problem. The following theorem is a *second order necessary optimality condition*

Unconstrained Optimization: 2nd Order Necessary Condition for Local Minimum

The above corollary is a *first order necessary optimality condition* for an unconstrained minimization problem. The following theorem is a *second order necessary optimality condition*

Theorem 5 *Suppose that $f(x)$ is twice continuously differentiable at $\bar{x} \in X$. If $\bar{x}$ is a local minimum, then $\nabla f(\bar{x}) = 0$ and $H(\bar{x})$ is positive semidefinite.*

Proof: From the first order necessary condition, $\nabla f(\bar{x}) = 0$. Suppose $H(\bar{x})$ is not positive semi-definite. Then there exists d such that $d^t H(\bar{x})d < 0$. We have:

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^t d + \frac{1}{2}\lambda^2 d^t H(\bar{x})d + \lambda^2 \|d\|^2 \alpha(\bar{x}, \lambda d)$$

$$= f(\bar{x}) + \frac{1}{2}\lambda^2 d^t H(\bar{x})d + \lambda^2 \|d\|^2 \alpha(\bar{x}, \lambda d),$$

where $\alpha(\bar{x}, \lambda d) \to 0$ as $\lambda \to 0$. Rearranging,

$$\frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda^2} = \frac{1}{2} d^t H(\bar{x})d + \|d\|^2 \alpha(\bar{x}, \lambda d).$$

Since $d^t H(\bar{x})d < 0$ and $\alpha(\bar{x}, \lambda d) \to 0$ as $\lambda \to 0$, $f(\bar{x} + \lambda d) - f(\bar{x}) < 0$ for all $\lambda > 0$ sufficiently small, yielding the desired contradiction. ∎

Unconstrained Optimization: Sufficient Condition for Local Minimum

Theorem 6 *Suppose that $f(x)$ is twice differentiable at $\bar{x}$. If $\nabla f(\bar{x}) = 0$ and $H(\bar{x})$ is positive definite, then $\bar{x}$ is a (strict) local minimum.*

Proof:

$$f(x) = f(\bar{x}) + \frac{1}{2}(x-\bar{x})^t H(\bar{x})(x-\bar{x}) + \|x-\bar{x}\|^2 \alpha(\bar{x}, x-\bar{x}).$$

Suppose that $\bar{x}$ is not a strict local minimum. Then there exists a sequence $x_k \to \bar{x}$ such that $x_k \neq \bar{x}$ and $f(x_k) \leq f(\bar{x})$ for all k. Define $d_k = \frac{x_k-\bar{x}}{\|x_k-\bar{x}\|}$. Then

$$f(x_k) = f(\bar{x}) + \|x_k-\bar{x}\|^2 \left(\frac{1}{2}d_k^t H(\bar{x}) d_k + \alpha(\bar{x}, x_k-\bar{x})\right) \; ,$$

and so

$$\frac{1}{2}d_k^t H(\bar{x}) d_k + \alpha(\bar{x}, x_k-\bar{x}) = \frac{f(x_k)-f(\bar{x})}{\|x_k-\bar{x}\|^2} \leq 0 \; .$$

Since $\|d_k\| = 1$ for any k, there exists a subsequence of $\{d_k\}$ converging to some point d such that $\|d\| = 1$. Assume without loss of generality that $d_k \to d$. Then

$$0 \geq \lim_{k\to\infty} d_k^t H(\bar{x}) d_k + \alpha(\bar{x}, x_k-\bar{x}) = \frac{1}{2}d^t H(\bar{x}) d,$$

which is a contradiction of the positive definiteness of $H(\bar{x})$. ∎

Steepest Descent Method (Uses Gradient Vector – 1st Order)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be real-valued function of n real variables

At any point x where gradient vector is nonzero, negative gradient, $-\nabla f(x)$, points downhill toward lower values of function f

In fact, $-\nabla f(x)$ is locally direction of steepest descent: function decreases more rapidly along direction of negative gradient than along any other

Steepest descent method: starting from initial guess x_0, successive approximate solutions given by

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),$$

where α_k is *line search* parameter that determines how far to go in given direction

Steepest Descent, continued

Given descent direction, such as negative gradient, determining appropriate value for α_k at each iteration is one-dimensional minimization problem

$$\alpha_k = \arg\min_{\alpha_k} f(x_k - \alpha_k \nabla f(x_k))$$

α_k is a hyperparameter

that can be solved by methods already discussed

Steepest descent method is very reliable: can always make progress provided gradient is nonzero

But method is myopic in its view of function's behavior, and resulting iterates can zigzag back and forth, making very slow progress toward solution

In general, convergence rate of steepest descent is only linear, with constant factor that can be arbitrarily close to 1

Example: Steepest Descent

Use steepest descent method to minimize

$$f(x) = 0.5x_1^2 + 2.5x_2^2$$

Gradient is given by

Gradient $\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix}$, Hessian $H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$

Taking $x_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$, we have $\nabla f(x_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$

Performing line search along negative gradient direction, i.e.,

$$\min_{\alpha_0} f(x_0 - \alpha_0 \nabla f(x_0)),$$

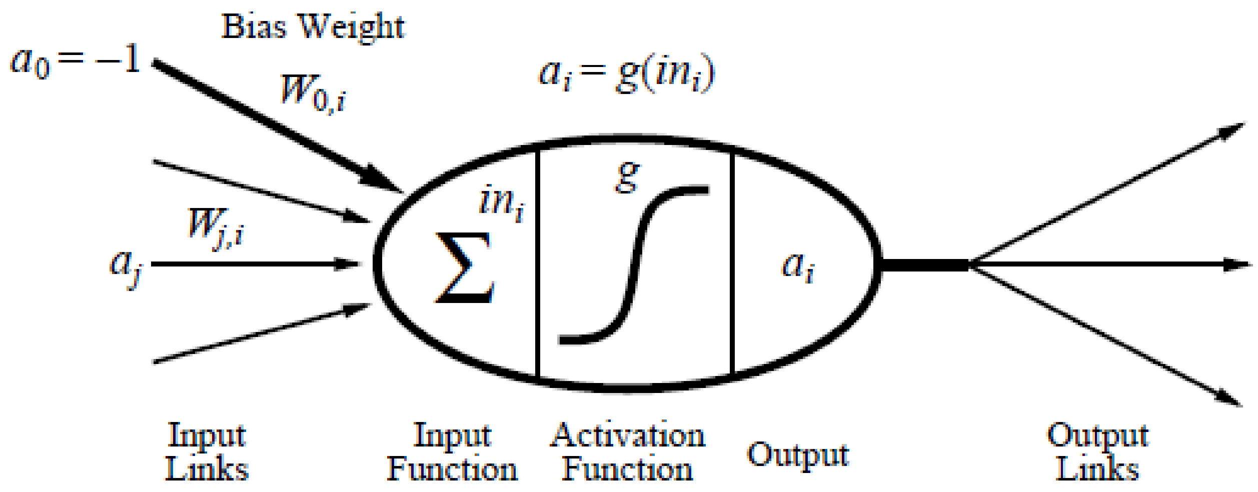
exact minimum along line is given by $\alpha_0 = 1/3$, so next approximation is

$$x_1 = \begin{bmatrix} 3.333 \\ -0.667 \end{bmatrix}$$

McCulloch-Pitts 'Neuron'

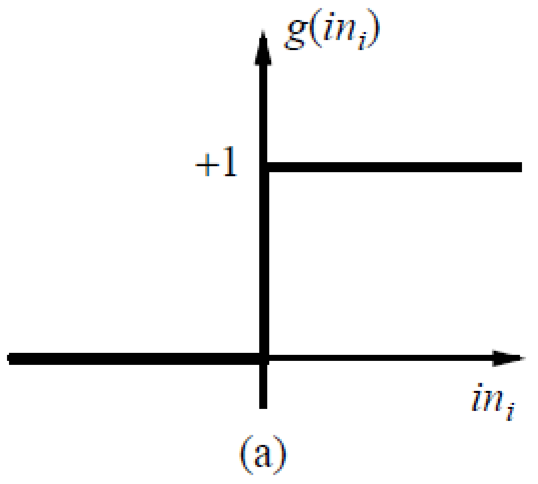
Output is a "squashed" linear function of the inputs:

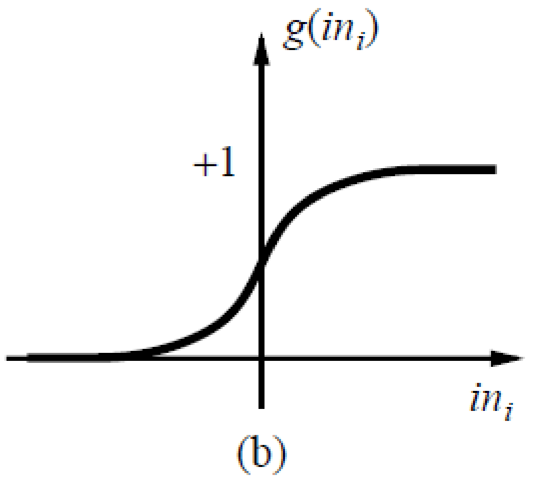
$$a_i \leftarrow g(in_i) = g\left(\Sigma_j W_{j,i} a_j\right)$$



A gross oversimplification of real neurons, but its purpose is to develop understanding of what networks of simple units can do

Activation Function *g*





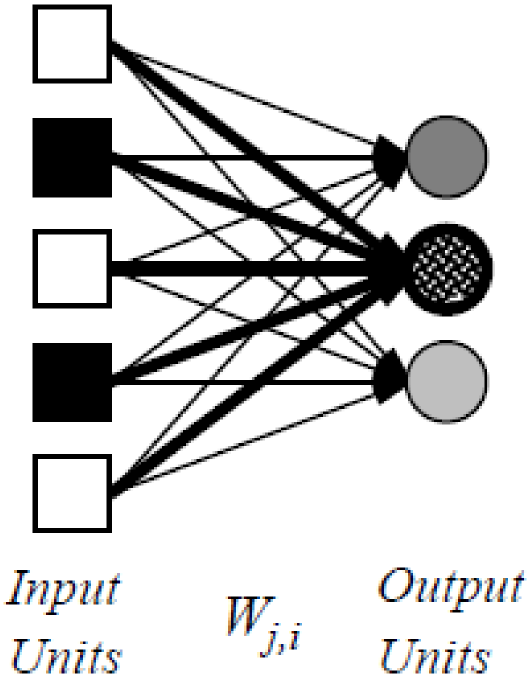
(a) is a step function or threshold function

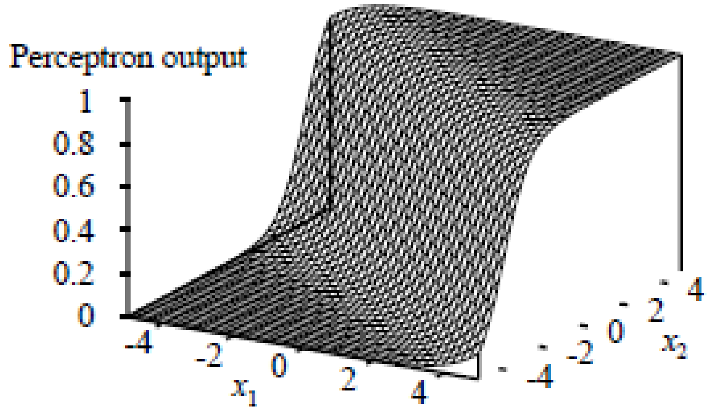
(b) is a sigmoid function $1/(1+e^{-x})$ $\quad \frac{dg(x)}{dx} = g(x)\{1-g(x)\}$

Changing the bias weight $W_{0,i}$ moves the threshold location

Single-Layer Perceptron

- Rosenblatt
  - *g* is step function
  - Can represent AND, OR, NOT. But not XOR





Gradient Descent for Single-Layer Perceptron

Learn by adjusting weights to reduce error on training set

The squared error for an example with input $\mathbf{x}$ and true output y is

$$E = \frac{1}{2}Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{W}}(\mathbf{x}))^2 \ ,$$

Perform optimization search by gradient descent:

$$\begin{aligned}\frac{\partial E}{\partial W_j} &= Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g(\Sigma_{j=0}^{n} W_j x_j)\right) \\ &= -Err \times g'(in) \times x_j\end{aligned}$$

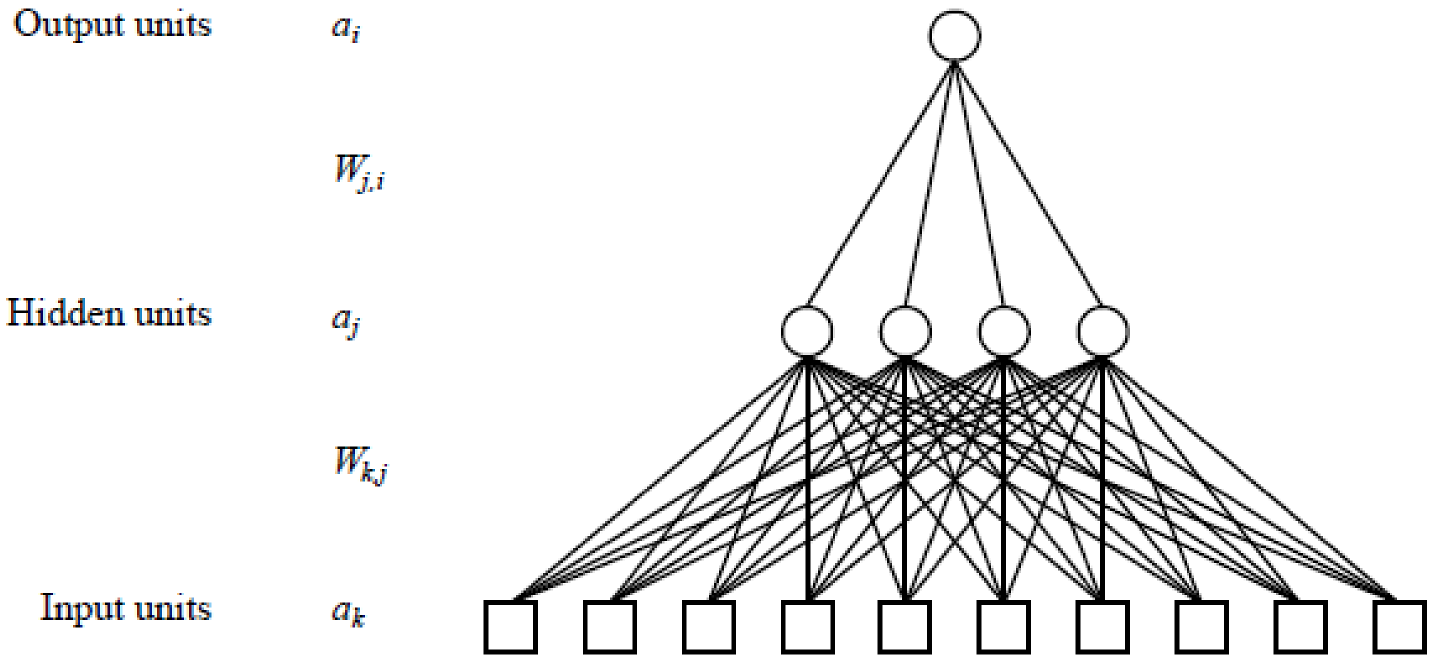
Simple weight update rule:

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

E.g., +ve error $\Rightarrow$ increase network output
$\Rightarrow$ increase weights on +ve inputs, decrease on -ve inputs

Multi-Layer Perceptron/Neural Network/Deep Learning

Layers are usually fully connected;
numbers of hidden units typically chosen by hand



Backpropagation Algorithm

- Rumelhart, Hinton and Williams

Output layer: same as for single-layer perceptron,

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

where $\Delta_i = Err_i \times g'(in_i)$

Hidden layer: back-propagate the error from the output layer:

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i \ .$$

Update rule for weights in hidden layer:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j \ .$$

- Two ways to implement weight updating
  1. BATCH UPDATES: sum (or average) updates across every sample *n*, then change the parameter values

$$w \leftarrow w + 2\alpha \sum_{n=1}^{N} \Delta^n a^n$$

  2. Stochastic/Online Update: update the parameters for each training case in turn, according to its own gradients

```
i. Randomly shuffle data in the training set
ii. for $n = 1$ to $N$ do
iii.     Update:
             $w \leftarrow w + 2\alpha \; \Delta^n a^n$
iv. end for
```

Shortcoming of Steepest Descent Method

Consider the unconstrained minimization problem:

$$\min_{x \in R^n} f : R^n \to R$$

Start with steepest descent :

The most rapid descent direction, $\nabla f(x)^T p$, for a given norm is:

$$\min_{p \in R^n} \nabla f(x)^T p \quad \text{subject to} \; \|p\| = 1$$

Consider Lagrangian $L = \nabla f(x)^T p + \lambda\left(p^T p - 1\right)$

$$\Rightarrow p = -\frac{\nabla f(x)}{\|\nabla f(x)\|}; \text{ negative gradient is the steepest gradient}$$

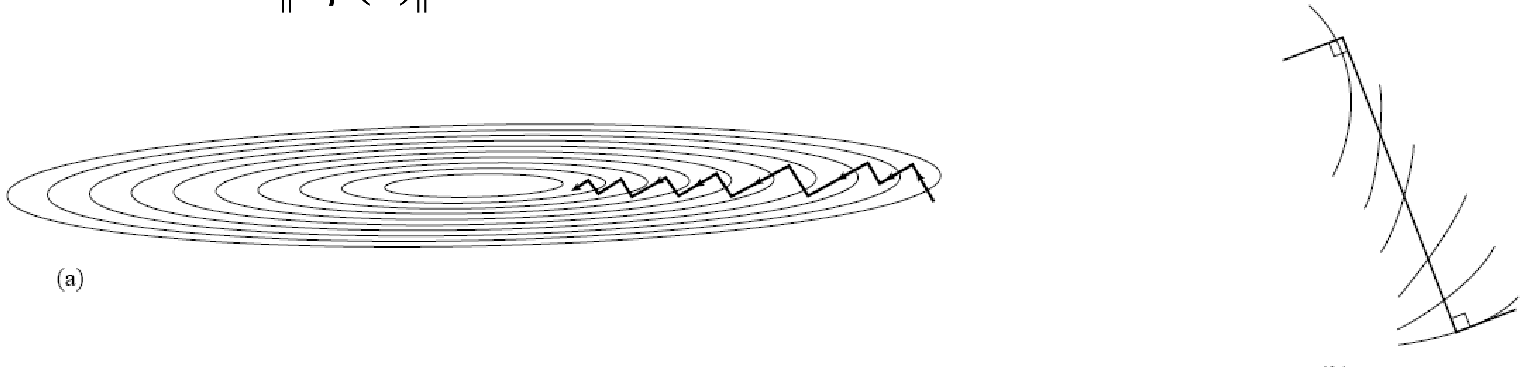


Figure . (a) Steepest descent method in a long, narrow "valley." While more efficient than the strategy of Figure 10.5.1, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient direction, perpendicular to the contour lines, and traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines.

Conjugate Gradient Method (Uses Hessian Matrix – 2nd Order)

Another method that does not require explicit second derivatives, and does not even store approximation to Hessian matrix, is *conjugate gradient* (CG) method

CG generates sequence of conjugate search directions, implicitly accumulating information about Hessian matrix

For quadratic objective function, CG is theoretically exact after at most n iterations, where n is dimension of problem

CG is effective for general unconstrained minimization as well

NOT IN SYLLABUS

II. **UNSUPERVISED LEARNING**

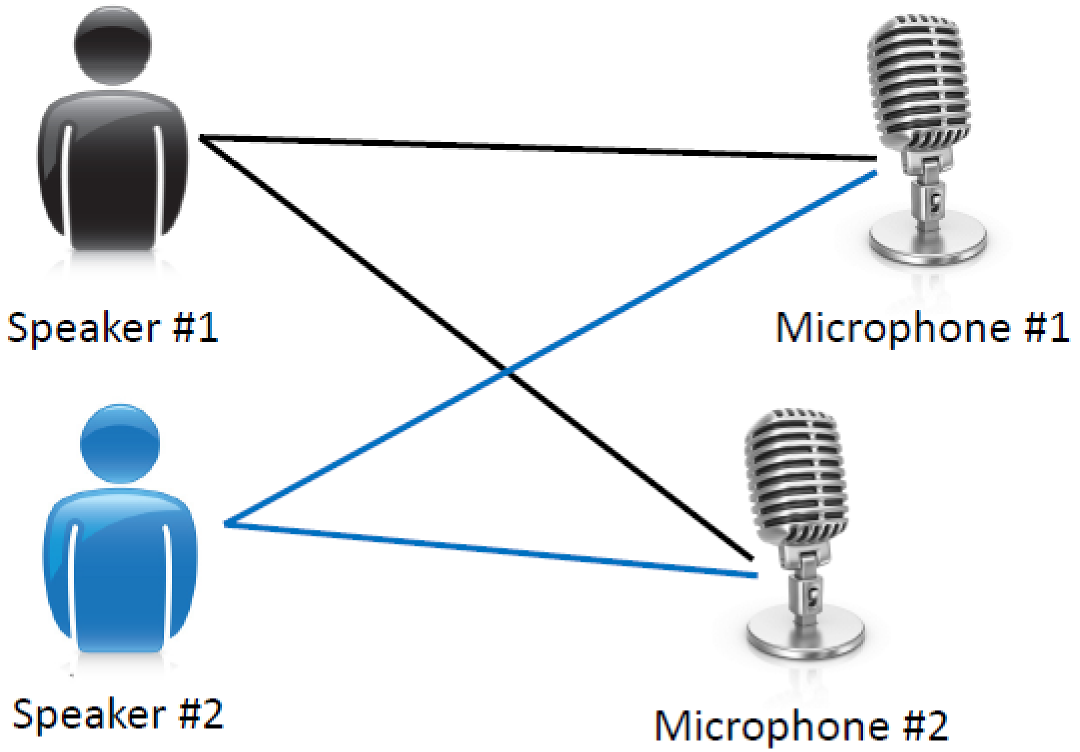
Unsupervised Learning

- Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.
- We can derive this structure by clustering the data based on relationships among the variables in the data.
- With unsupervised learning there is no feedback (**no label**) based on the prediction results.

Unsupervised Learning

- **Example:**
- Non-clustering : The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a cocktail party). Differentiate using **Support Vector Machine (SVM)** – which could also be used in supervised learning.
- Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on. Dimension Reduction using Principal Component Analysis (**PCA**).

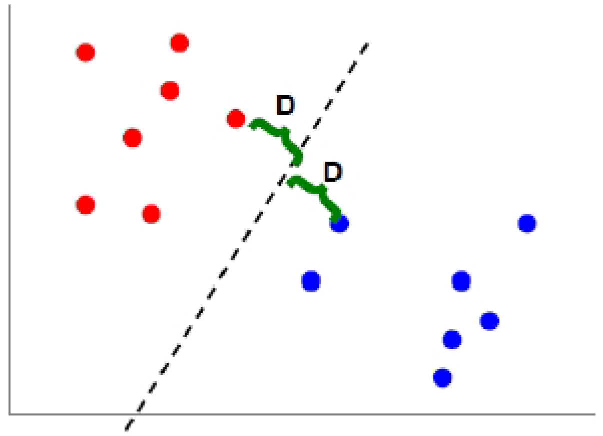
Cocktail party problem



3. Support Vector Machines: Constrained Optimization

Max-Margin Classification

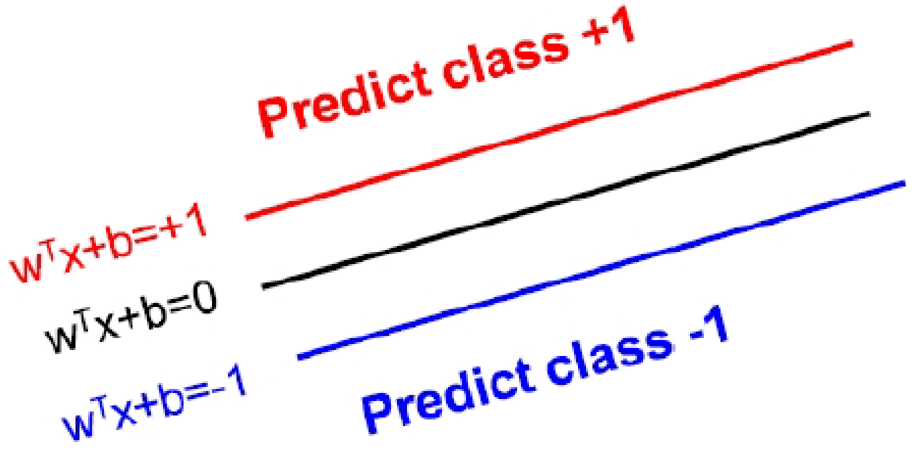
- Instead of fitting all the points, focus on boundary points
- Aim: learn a boundary that leads to the largest margin (buffer) from points on both sides



- Why: intuition; theoretical support; and works well in practice
- Subset of vectors that support (determine boundary) are called the support vectors

Linear Support Vector Machine (SVM)

- Max margin classifier: inputs in margin are of unknown class

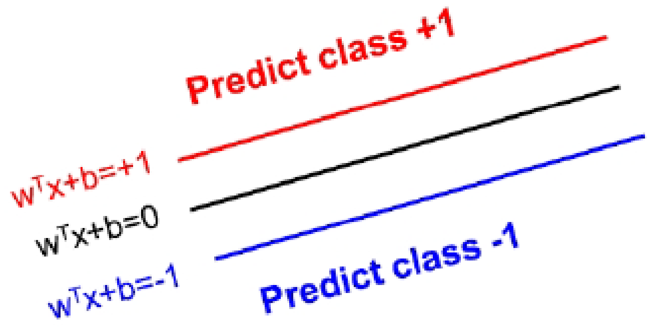


$$y = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 1 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} + b \leq -1 \\ \text{Undefined} & \text{if } -1 \leq \mathbf{w}^T\mathbf{x} + b \leq 1 \end{cases}$$

- Can write above condition as:

$$(\mathbf{w}^T\mathbf{x} + b)y \geq 1$$

Geometry of Problem



- The vector **w** is orthogonal to the $+1$ plane.
  If **u** and **v** are two points on that plane, then

$$\mathbf{w}^T(\mathbf{u}-\mathbf{v})=0$$

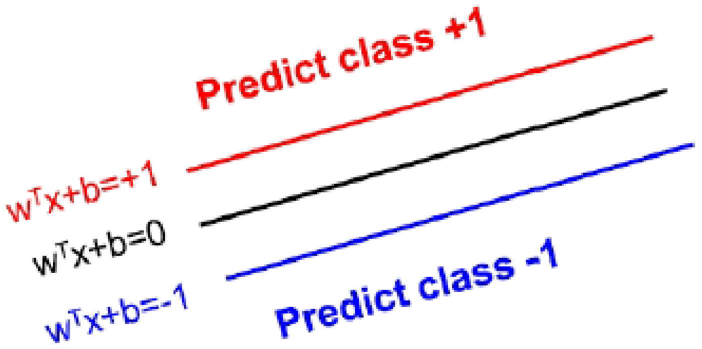
- Same is true for -1 plane
- Also: for point $\mathbf{x}_+$ on $+1$ plane and $\mathbf{x}_-$ nearest point on -1 plane:

$$\mathbf{x}_+=\lambda\mathbf{w}+\mathbf{x}_-$$

Computing the Margin

- Also: for point $\mathbf{x}_+$ on $+1$ plane and $\mathbf{x}_-$ nearest point on -1 plane:

$$\mathbf{x}_+ = \lambda \mathbf{w} + \mathbf{x}_-$$



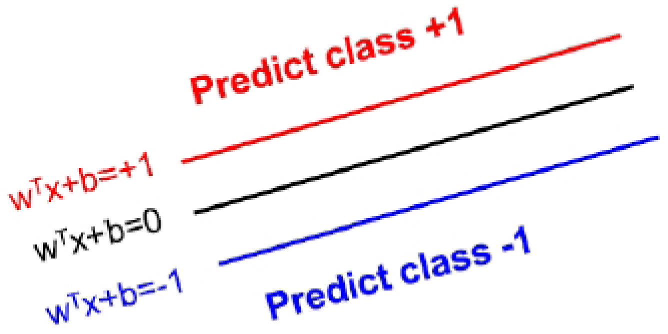
$$\begin{aligned}
\mathbf{w}^T\mathbf{x}_+ + b &= 1 \\
\mathbf{w}^T(\lambda\mathbf{w} + \mathbf{x}_-) + b &= 1 \\
\mathbf{w}^T\mathbf{x}_- + b + \lambda\mathbf{w}^T\mathbf{w} &= 1 \\
-1 + \lambda\mathbf{w}^T\mathbf{w} &= 1
\end{aligned}$$

Therefore

$$\lambda = \frac{2}{\mathbf{w}^T\mathbf{w}}$$

Computing the Margin

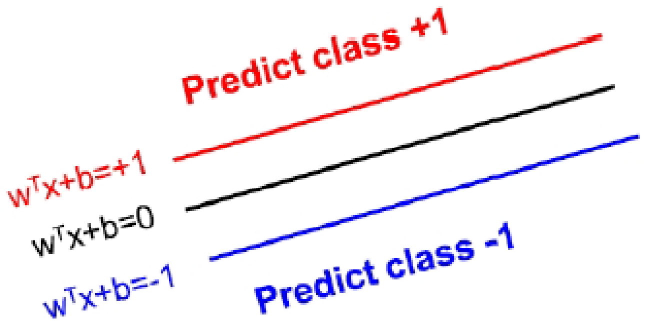
- Define the margin M to be the distance between the $+1$ and -1 planes
- We can now express this in terms of $\mathbf{w}$ to maximize the margin we minimize the length of $\mathbf{w}$



$$
\begin{aligned}
M &= ||\mathbf{x}_+ - \mathbf{x}_-|| \\
&= ||\lambda \mathbf{w}|| = \lambda \sqrt{\mathbf{w}^T \mathbf{w}} \\
= 2\frac{\sqrt{\mathbf{w}^T \mathbf{w}}}{\mathbf{w}^T \mathbf{w}} &= \frac{2}{\sqrt{\mathbf{w}^T \mathbf{w}}} = \frac{2}{||\mathbf{w}||}
\end{aligned}
$$

Margin-Based Classifier

- We can search for the optimal parameters ($\mathbf{w}$ and b) by finding a solution that:
  1. Correctly classifies the training examples: $\{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$
  2. Maximizes the margin (same as minimizing $\mathbf{w}^T\mathbf{w}$)



$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

$$s.t.\forall i \quad (\mathbf{w}^T\mathbf{x}^{(i)} + b)t^{(i)} \geq 1,$$

- This is called the primal formulation of Support Vector Machine (SVM)
- Can optimize via projective gradient descent, etc.
- Apply Lagrange multipliers: formulate equivalent problem

Constrained Optimization Problem

Equality and Inequality Constraints

$$\min f(x)$$

$$\text{s.t. } g_i(x) \le 0, \quad i = 1, \ldots, k$$

$$h_i(x) = 0, \quad i = 1, \ldots, m$$

Define the generalized Lagrangian

$$L(x, \mu, \lambda) = f(x) + \sum_{i=1}^{k} \mu_i g_i(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

where μ_i and λ_i are the Lagrange multipliers.

Constrained Optimization Problem

- First Order Necessary Conditions

Karush-Kuhn Tucker Conditions:

If x^* is a primal (local) optimal and $\left(\mu^*, \lambda^*\right)$ are dual (local) optimal, the following conditions must be satisfied:

1. (Lagrangian Stationarity) $\nabla_x L\left(x^*, \mu^*, \lambda^*\right)=0$
2. (Complementary Slackness) $\mu_i^* g_i\left(x^*\right) = 0,\ i = 1, \ldots, m$
3. (Primal Feasibility) $g_i\left(x^*\right) \leq 0,\ i = 1, \ldots, m$ and $h_i\left(x^*\right) = 0,\ i = 1, \ldots, k$
4. (Dual Feasibility) $\mu_i^* \geq 0,\ i = 1, \ldots, m$

- Sufficient Condition

If strong duality holds, then above conditions are also sufficient.

Example : Constrained Optimization

Consider the problem

minimize $2x_1^2 + 2x_1x_2 + x_2^2 - 10x_1 - 10x_2$

subject to $x_1^2 + x_2^2 \le 5$

$$3x_1 + x_2 \le 6$$

The first order necessary conditions, in addition to the constratins, are

$$4x_1 + 2x_2 - 10 + 2\mu_1 x_1 + 3\mu_2 = 0$$

$$2x_1 + 2x_2 - 10 + 2\mu_1 x_2 + \mu_2 = 0$$

$$\mu_1 \ge 0, \quad \mu_2 \ge 0$$

$$\mu_1\left(x_1^2 + x_2^2 - 5\right) = 0$$

$$\mu_2\left(3x_1 + x_2 - 6\right) = 0$$

We have to try various combinations of active constraints and check the signs of the resulting Lagrange multipliers μ_1 and μ_2. We shall assume the first constraint is active:

$$\left.\begin{aligned} 4x_1 + 2x_2 - 10 + 2\mu_1 x_1 &= 0 \\ 2x_1 + 2x_2 - 10 + 2\mu_1 x_2 &= 0 \\ x_1^2 + x_2^2 &= 5 \end{aligned}\right\} \Rightarrow x_1 = 1,\ x_2 = 2,\ \mu_1 = 1$$

and $3x_1 + x_2 = 5,$ thus second constraint is satisfied. Since $\mu_1 > 0$, we conclude solution satisfies the first-order necessary KKT conditions.

Constrained Optimization Problem (Primal-Dual)

- Primal Objective

$$\theta_P(x) = \max_{\mu,\lambda:\mu_i \geq 0, \forall i} L(x,\mu,\lambda) = \max_{\mu,\lambda:\mu_i \geq 0, \forall i} f(x) + \sum_{i=1}^{k} \mu_i g_i(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

$$= f(x) + \max_{\mu,\lambda:\mu_i \geq 0, \forall i} \sum_{i=1}^{k} \mu_i g_i(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

$$= \begin{cases} f(x), & \text{if } x \text{ is primal feasible, i.e. } g_i(x) \leq 0,\ h_i(x) = 0 \\ \infty \quad , & \text{otherwise} \end{cases}$$

- <u>Primal Problem</u>

$$p^* = \min_x \theta_P(x) = \min_x \max_{\mu,\lambda:\mu_i \geq 0} L(x,\mu,\lambda)$$

$$= \min_x \max_{\mu,\lambda:\mu_i \geq 0 \forall i} f(x) + \sum_{i=1}^{k} \mu_i g_i(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

$$= \text{optimal value of the primal problem}$$

NOT IN SYLLABUS

Constrained Optimization Problem (Primal-Dual)

- Dual Objective

$$\theta_D(\mu,\lambda) = \min_x L(x,\mu,\lambda)$$

$$= \min_x f(x) + \sum_{i=1}^{k} \mu_i g_i(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

- <u>Dual Problem</u>

$$d^* = \max_{\mu,\lambda:\mu_i \geq 0,\forall i} \theta_D(\mu,\lambda)$$

$$= \max_{\mu,\lambda:\mu_i \geq 0,\forall i} \min_x L(x,\mu,\lambda)$$

$=$ optimal value of the dual objective

(μ,λ) are dual feasible if $\mu_i \geq 0, i = 1,\ldots,m$

NOT IN SYLLABUS

Constrained Optimization Problem (Primal-Dual)

- Lemma 1

If (μ,λ) are dual feasible, then $\theta_D(\mu,\lambda) \le p^*$

Proof: $\theta_D(\mu,\lambda) = \min_x L(x,\mu,\lambda)$

$$\le L(x^*,\mu,\lambda) = f(x^*) + \sum_{i=1}^{k} \mu_i g_i(x^*) + \sum_{i=1}^{m} \lambda_i h_i(x^*) \;\ldots(*)$$

$$\le f(x^*) = p^*$$

In $(*)$, last 2 terms are ≤ 0 when x^* is primal feasible & (μ,λ) are dual feasible.

- Lemma 2 (follows from Lemma 1) – Weak Duality

$$d^* = \max_{\mu,\lambda:\mu_i\ge 0\forall i} \theta_D(x) \le p^*$$

- Lemma 3 – Strong Duality

For any primal and dual problems which satisfy certain constraint qualifications, e.g. convex problems or Slater's condition, then $d^* = p^*$.

NOT IN SYLLABUS

4. Dimension Reduction: Principal Component Analysis

Motivation

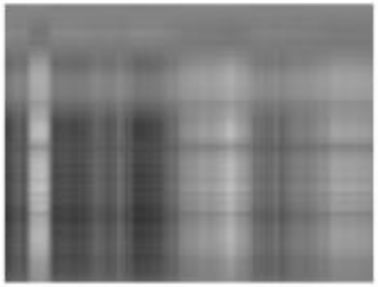
- High-dimensional data, such as images, is hard to analyse, interpret and visualize
- Requires higher storage and computational cost
- Dimensions in high-dimensional data is often correlated, i.e. data can be represented in lower-dimensional structre
- Minimum Variance Formulation & Minimum Error Reconstruction
- Key concept is based on Eigenvalue Decomposition of Input Covariance Matrix
- Dimensionality Reduction exploits features selection (whether correlated to output) and feature derivation (via applying transform)

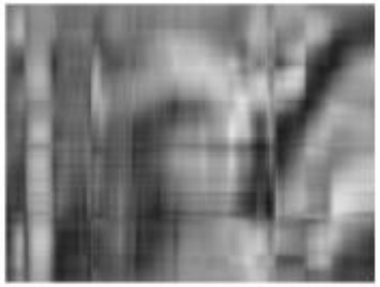
Continuous Underlying Variables In Images

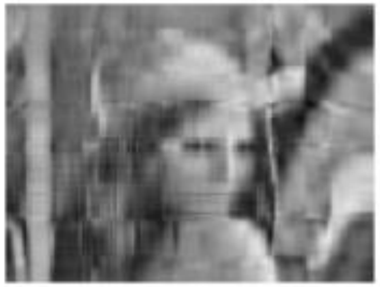
- What are the intrinsic latent dimensions in these two datasets?

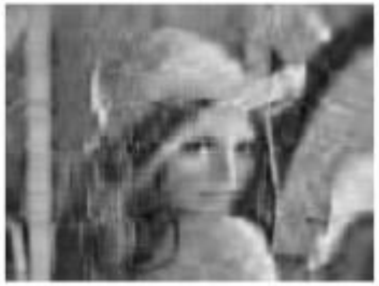
- How can we find these dimensions from the data?

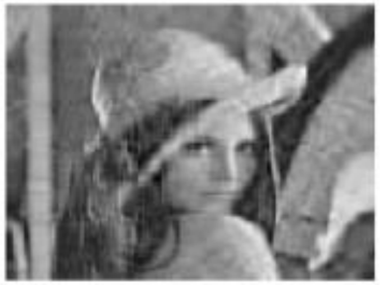
Application: Image Compression

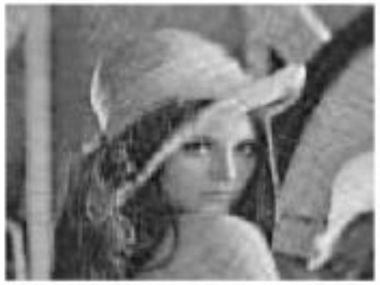

(a) 1 principal component

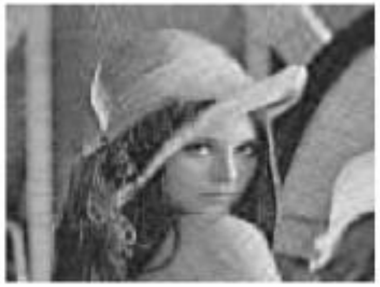

(b) 5 principal component

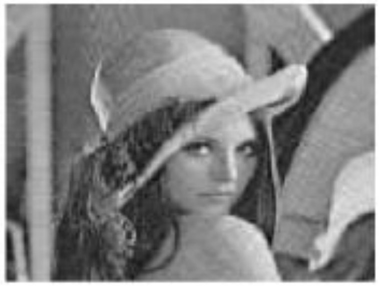

(c) 9 principal component


(d) 13 principal component


(e) 17 principal component


(f) 21 principal component


(g) 25 principal component


(h) 29 principal component

Principal Components Analysis (PCA) (also known as Karhunen-Loeve Transform)

- PCA: most popular instance of second main class of unsupervised learning methods, projection methods, aka dimensionality-reduction methods
- Aim: find a small number of "directions" in input space that explain variation in input data; re-represent data by projecting along those directions
- Important assumption: variation contains information
- Data is assumed to be continuous:
  - linear relationship between data and the learned representation

Principal Components Analysis (PCA)

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization
  - Preprocessing
  - Modeling – prior for new data
  - Compression

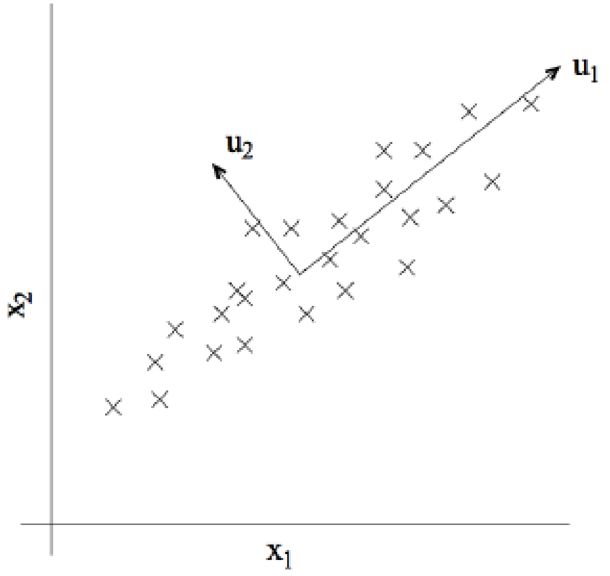
Principal Components Analysis (PCA)

- Assume that set of training data has N vectors, $\{\mathbf{x}_n\}_{n=1}^N$, of dimensionality D, so $\mathbf{x}_i \in \mathbb{R}^D$
- Aim to reduce dimensionality:
  - linearly project to a much lower dimensional space, $M << D$:

$$\mathbf{x} \approx U\mathbf{z} + \mathbf{a}$$

    where U a $D \times M$ matrix and $\mathbf{z}$ a M-dimensional vector

- Search for orthogonal directions in space with the highest variance
  - project data onto this subspace
- Structure of data vectors is encoded in sample covariance



Eigenvalues and Eigenvectors

Given an $n \times n$ matrix A representing a linear transformation on an n-dimensional vector space, a scalar λ is an *eigenvalue* and a nonzero vector x is the corresponding *eigenvector* if they satisfy

$$Ax = \lambda x$$

This eigenvector x is also called the *right* eigenvector.

Define a nonzero vector y, called the *left* eigenvector, such that

$$y^T A = \lambda y^T$$

But here, we consider only the right eigenvectors.

Characteristic Polynomial

Equation $Ax = \lambda x$ equivalent to

$$(A - \lambda I)x = o,$$

which has nonzero solution x if, and only if, its matrix is singular

Eigenvalues of A are roots λ_i of *characteristic polynomial*

$$\det(A - \lambda I) = 0$$

in λ of degree n

Fundamental Theorem of Algebra implies that $n \times n$ matrix A always has n eigenvalues, but they may be neither real nor distinct

Complex eigenvalues of real matrix occur in complex conjugate pairs: if $\alpha + i\beta$ is eigenvalue of real matrix, then so is $\alpha - i\beta$, where $i = \sqrt{-1}$

Example: Characteristic Polynomial

Consider characteristic polynomial of previous example matrix:

$$\det\left(\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) =$$

$$\det\left(\begin{bmatrix} 3-\lambda & -1 \\ -1 & 3-\lambda \end{bmatrix}\right) =$$

$$(3-\lambda)(3-\lambda) - (-1)(-1) = \lambda^2 - 6\lambda + 8 = 0,$$

so eigenvalues given by

$$\lambda = \frac{6 \pm \sqrt{36-32}}{2},$$

or

$$\lambda_1 = 2, \qquad \lambda_2 = 4$$

Examples: Eigenvalues and Eigenvectors

1. $\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$:

$$\lambda_1 = 1,\ \boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \lambda_2 = 2,\ \boldsymbol{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

2. $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$:

$$\lambda_1 = 1,\ \boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \lambda_2 = 2,\ \boldsymbol{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

3. $\boldsymbol{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$:

$$\lambda_1 = 2,\ \boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \lambda_2 = 4,\ \boldsymbol{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Singular Value Decomposition

Singular value decomposition (SVD) of $m \times n$ matrix $\boldsymbol{A}$ has form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ is $m \times m$ orthogonal matrix, $\boldsymbol{V}$ is $n \times n$ orthogonal matrix, and $\boldsymbol{\Sigma}$ is $m \times n$ diagonal matrix, with

$$\sigma_{ij} = \begin{cases} 0 & \text{for } i \neq j \\ \sigma_i \geq 0 & \text{for } i = j \end{cases}$$

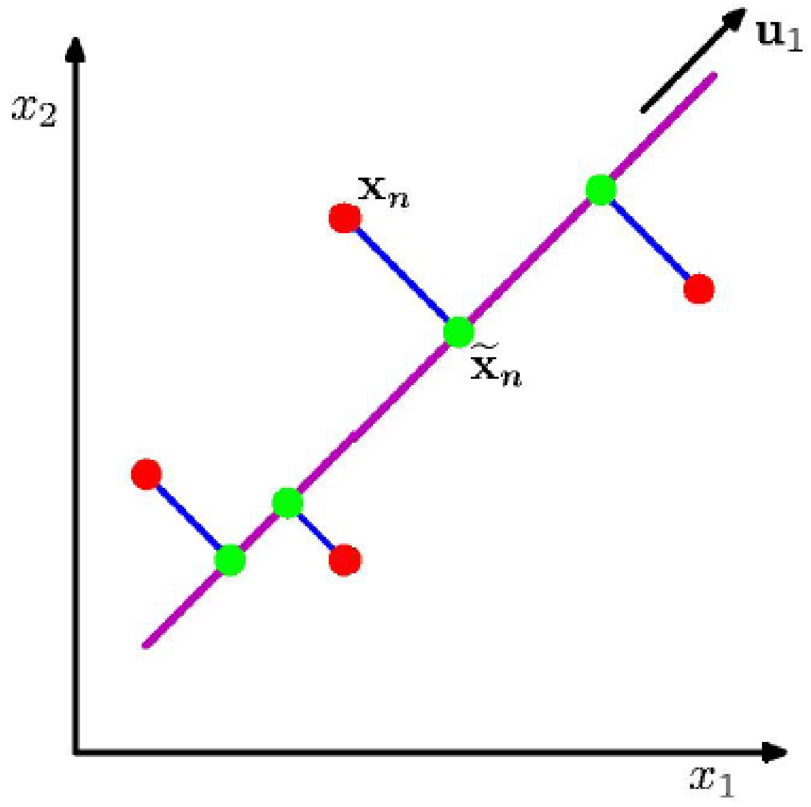
Diagonal entries σ_i, called *singular values* of $\boldsymbol{A}$, usually ordered so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$

Columns $\boldsymbol{u}_i$ of $\boldsymbol{U}$ and $\boldsymbol{v}_i$ of $\boldsymbol{V}$ called left and right *singular vectors*

For square symmetric matrix, $A = A^T$, $m=n$, $U = V$. $u_i=v_i$, $i=1,\ldots n$, and σ_i is the respective eigenvalues of A. So, $A=U\Sigma U^T$.

Derivation of PCA

- Two views/derivations:
  - Maximize variance (scatter of green points)
  - Minimize error (red-green distance per datapoint)



PCA: Minimum Variance Formulation

Goal: Project data onto space having dimensionality $M < D$ while maximizing variance of projected data.

- Consider a data set of observations $\{x_n\}$ where $n = 1, \ldots, N$.
- Each x_n is a Euclidean variable with dimensionality D.
- Assume projecting onto a one-dimensional space (M = 1).
- Define the direction of this space using $\mathbf{u}_1$.
- Assume $\mathbf{u}_1$ is a unit vector ($\mathbf{u}_1^T \mathbf{u}_1 = 1$).
- The mean of the projected data is $\mathbf{u}_1^T \bar{x}$ where $\bar{x}$ is the sample set mean

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

- The variance of the projected data is given by

$$\frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{u}_1^T x_n - \mathbf{u}_1^T \bar{x} \right)^2 = \mathbf{u}_1^T S \mathbf{u}_1$$

where S is the covariance matrix of the data, *S* is square and symmetric matrix

$$S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T.$$

PCA: Minimum Variance Formulation

- To maximize the variance, we solve the following constrained problem

$$\underset{\mathbf{u}_1}{\text{maximize}} \quad \mathbf{u}_1^T S \mathbf{u}_1 \quad \text{s.t.} \quad \mathbf{u}_1^T \mathbf{u}_1 = 1$$

- The Lagrangian of this problem is given by

$$\mathcal{L}(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^T S \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T \mathbf{u}_1).$$

- Differentiating with respect to $\mathbf{u}_1$, we have a stationary point when

$$S\mathbf{u}_1 = \lambda_1 \mathbf{u}_1.$$

- By left-multiplying by $\mathbf{u}_1$ and using $\mathbf{u}_1^T \mathbf{u}_1 = 1$, we have

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1.$$

- Thus, the maximum variance will occur when we set $\mathbf{u}_1$ to the eigenvector having the largest eigenvalue λ_1.

- Additional principal components can be defined in an incremental fashion.
- A similar problem can be formed for the minimum error formulation.
  - Solution is in terms of the $D - M$ smallest eigenvalues of the eigenvectors that are orthogonal to the principal subspace.

PCA : Minimizing Reconstruction Error (Compression)

- We can think of PCA as projecting the data onto a lower-dimensional subspace
- One derivation is that we want to find the projection such that the best linear reconstruction of the data is as close as possible to the original data

$$J(\mathbf{u}, \mathbf{z}, \mathbf{b}) = \sum_n ||\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}||^2$$

where

$$\tilde{\mathbf{x}}^{(n)} = \sum_{j=1}^{M} z_j^{(n)} \mathbf{u}_j + \sum_{j=M+1}^{D} b_j \mathbf{u}_j$$

- Objective minimized when first M components are the eigenvectors with the maximal eigenvalues

Coefficients of basis u_j given by : $z_j^{(n)} = \mathbf{u}_j^T \mathbf{x}^{(n)}; \quad b_j = \bar{\mathbf{x}}^T \mathbf{u}_j$

Standard PCA

- Algorithm: to find M components underlying D-dimensional data
  1. Select the top M eigenvectors of S (data covariance matrix):

     $$S = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T = U\Sigma U^T \approx U_{1:M}\Sigma_{1:M}U_{1:M}^T$$

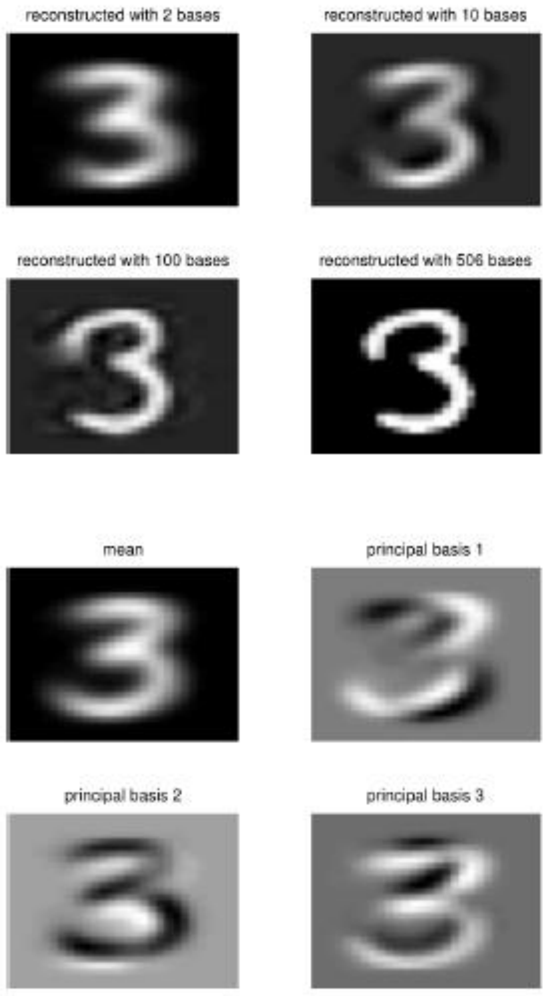
     where U is orthogonal, columns are unit-length eigenvectors

     $$U^TU = UU^T = 1$$

     and Σ is a matrix with eigenvalues on the diagonal, representing the variance in the direction of each eigenvector
  2. Project each input vector $\mathbf{x}$ into this subspace, e.g.,

     $$z_j = \mathbf{u}_j^T\mathbf{x}; \qquad \mathbf{z} = U_{1:M}^T\mathbf{x}$$

Application to Digits


reconstructed with 2 bases
reconstructed with 10 bases
reconstructed with 100 bases
reconstructed with 506 bases
mean
principal basis 1
principal basis 2
principal basis 3