Machine Learning and Music (IFT6080 Winter 08)
Prof. Douglas Eck, Université de Montréal

These slides follow closely the (English) course textbook
Pattern Recognition and Machine Learning
by Christopher Bishop

Goals of the course

- Understand basic concepts behind machine learning algorithms
  Prerequisites: Common sense
- Understand some elements of learning theory
  Prerequisites: Probability, statistics, linear algebra
- Implement and use machine learning algorithms
  Prerequisites: Algorithms, programming, numerical analysis

What is machine learning?

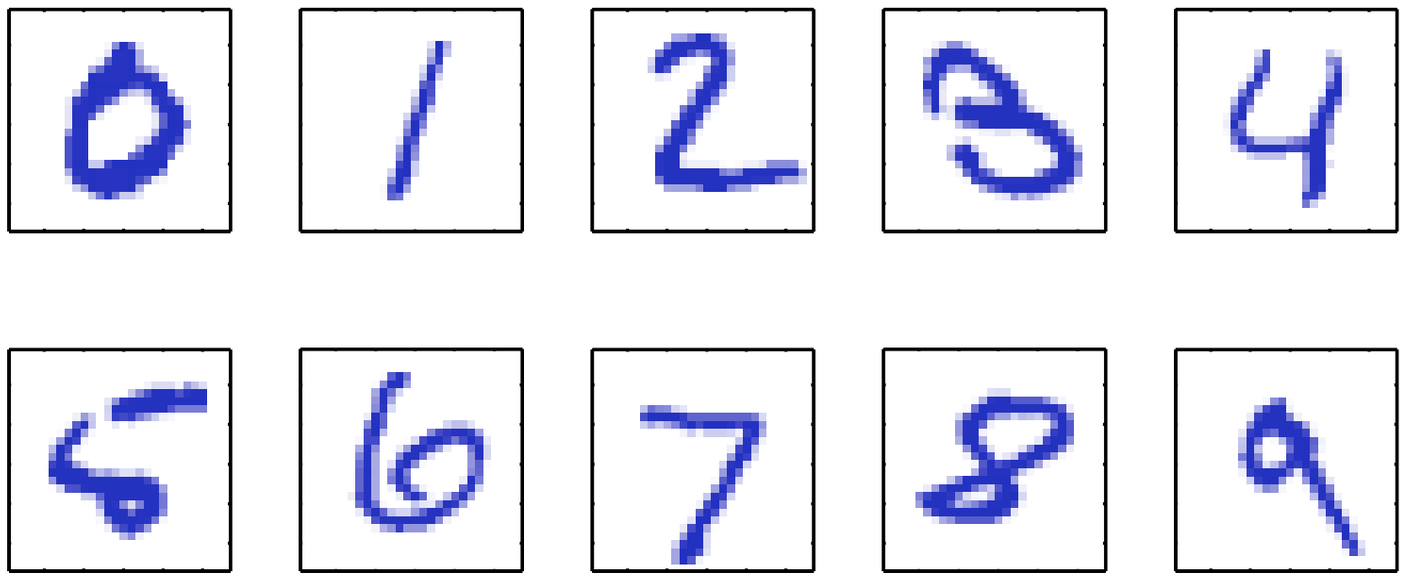
- Automatic discovery of regularities in data.
- Algorithms and techniques that allow computers to "learn". The major focus is to extract information from data automatically, by computational and statistical methods
- Applications: natural language processing, search engines, medical diagnosis, bioinformatics, stock market analysis, game playing and robot locomotion.
- http://en.wikipedia.org/wiki/Machine_learning

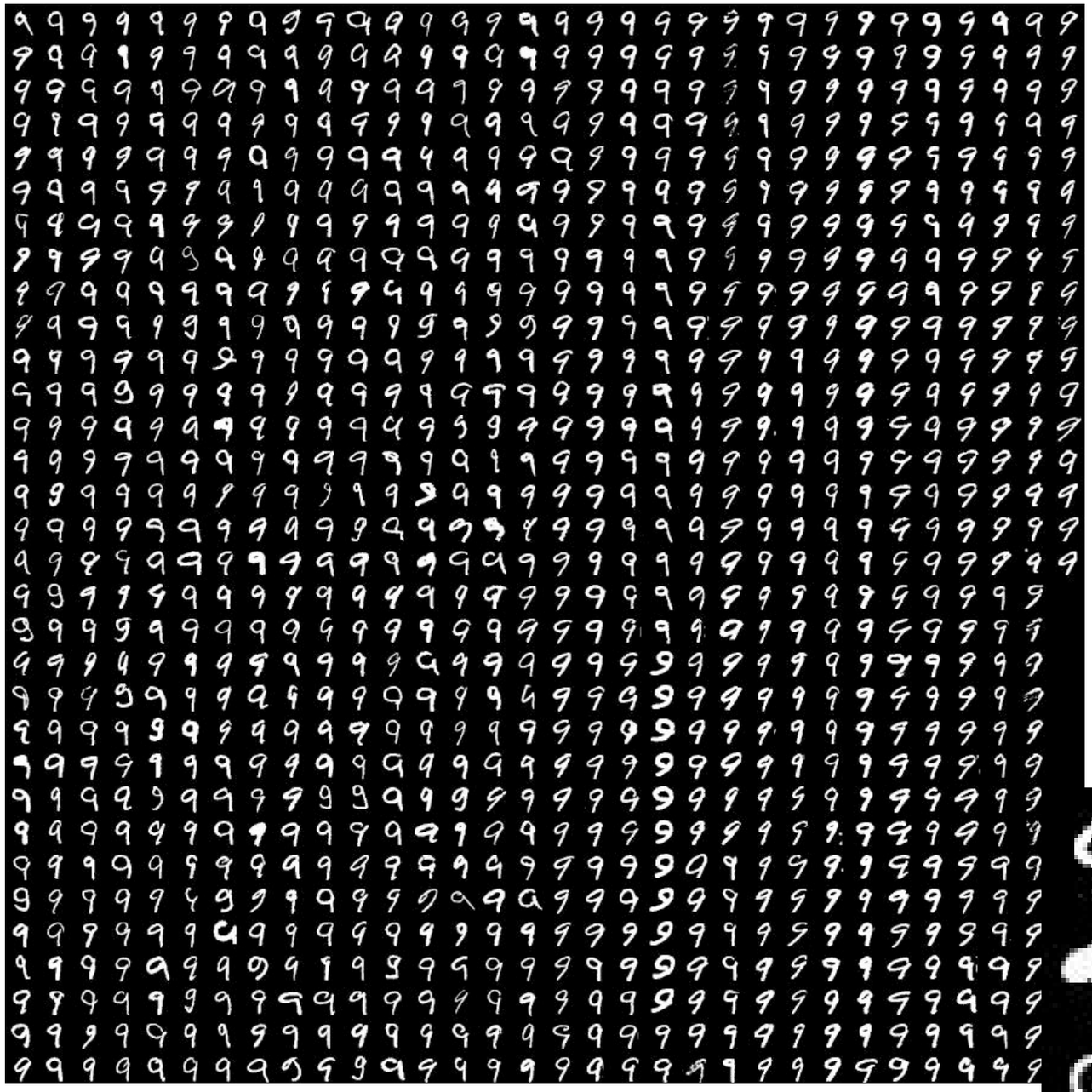
Relation to "intelligence"

- Learning is fundamental characteristic of human intelligence
- To learn is to *change for the better*
- One way to measure change is in terms of behavior of organism in new but similar situations
- Generalization is key: it is *easy* to learn by heart, *difficult* to learn general-purpose strategies
- Useful distinction: *innate* versus *acquired* knowledge (for us: priors versus data)

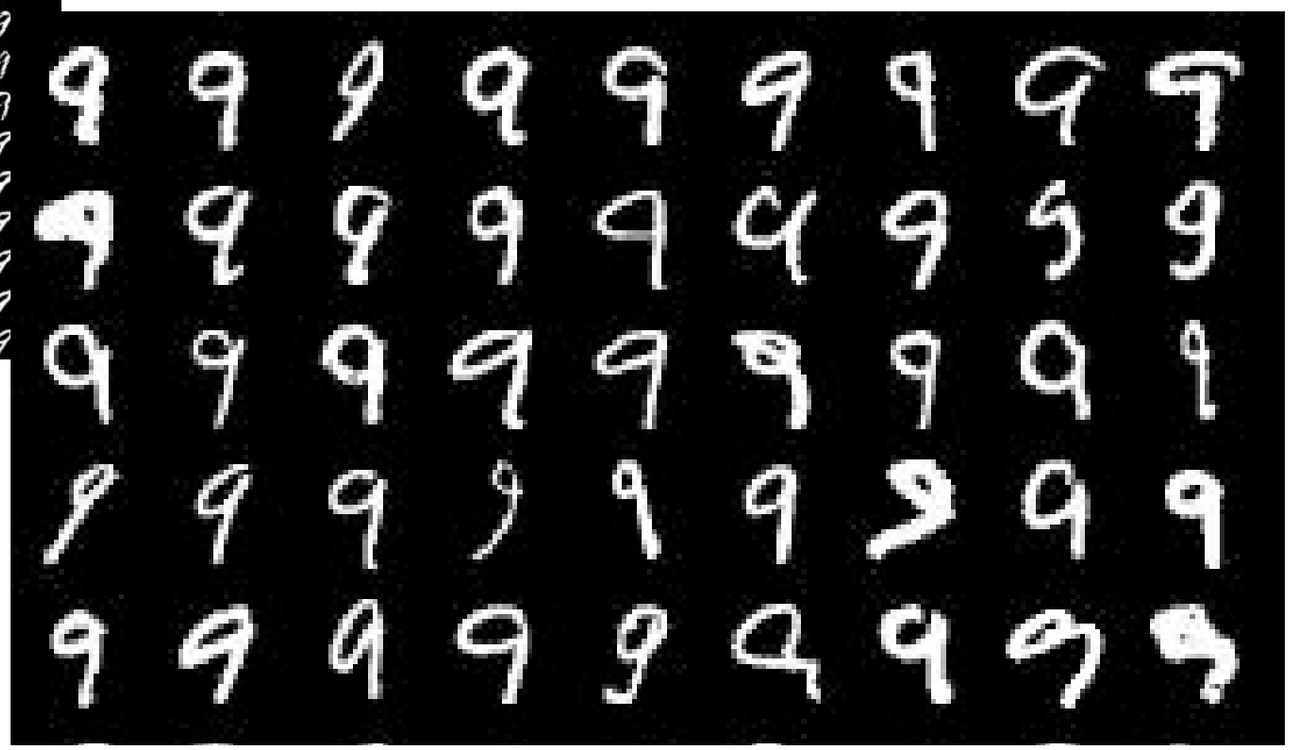
Example: Digit Recognition

- Example: 28 x 28 pixel image as vector **x**, |x|=78
- Build machine able to identify digit {0,1,...,9} as output





- Nontrivial: rules or heuristics yield poor results



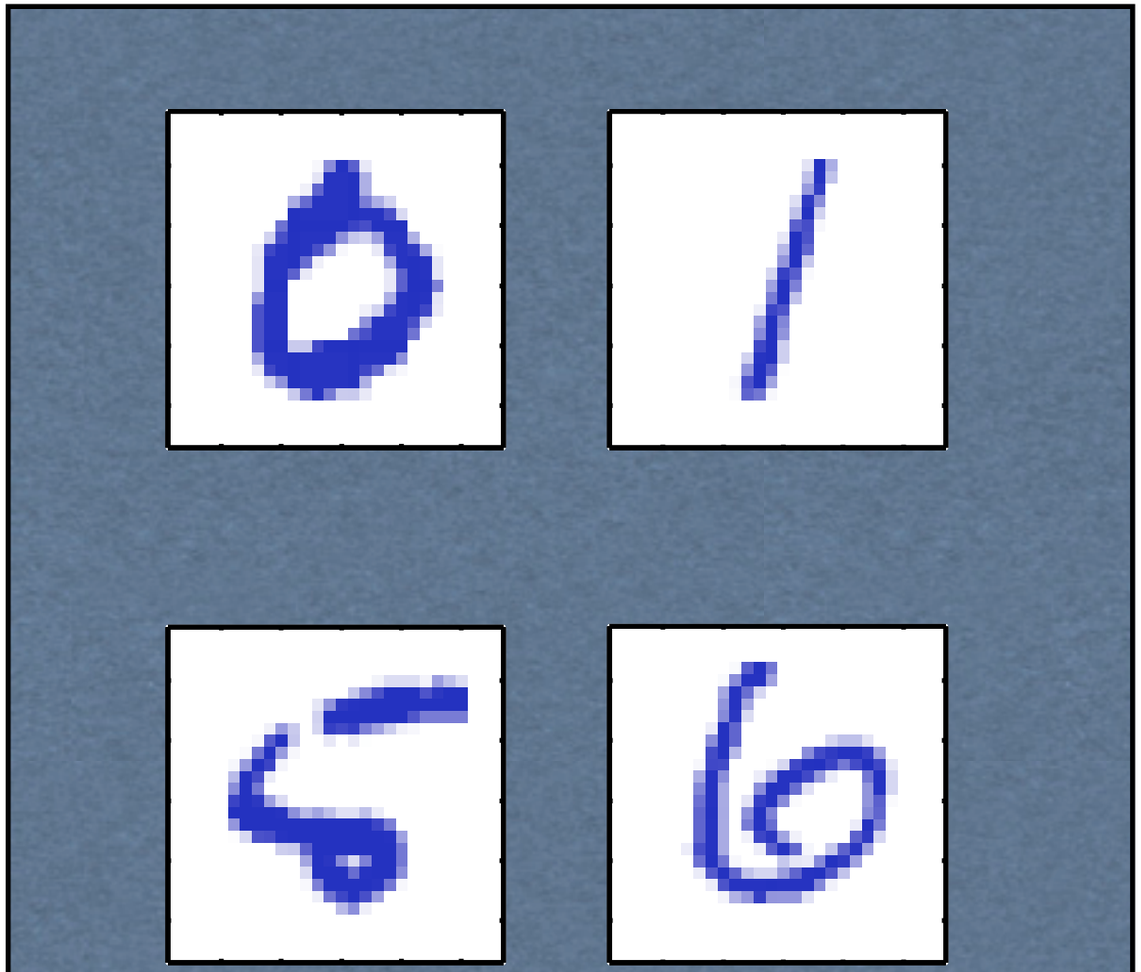
- http://yann.lecun.com/exdb/mnist/
- http://www.cs.toronto.edu/~roweis/data.html

Basic terminology

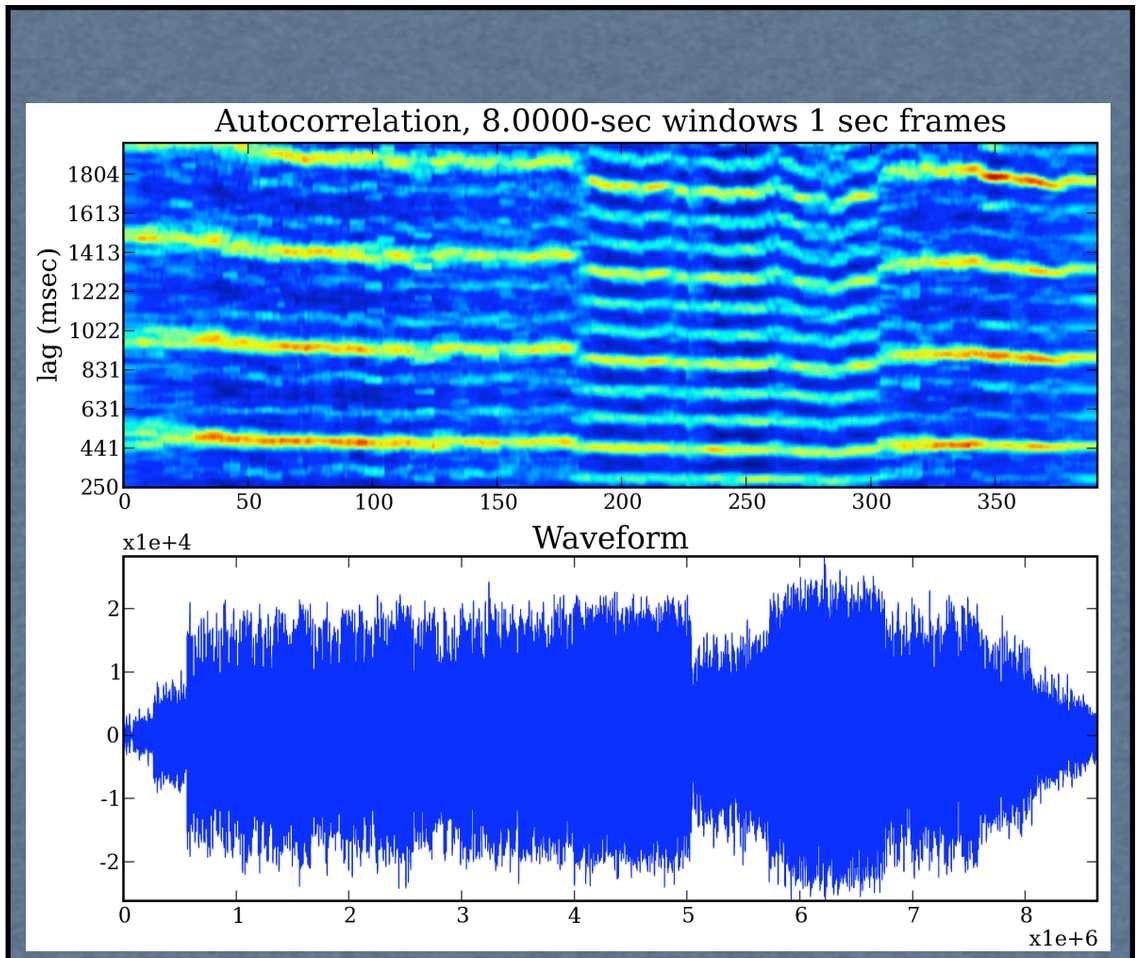
- Training Set: N digits $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
- Target Vector: Unique vector t for each target digit
- Learned Function: y(x)
- Training Phase: Process for determining y(x)
- Test Set: Some digit images not found in training set

Feature extraction

Transform original input variables into some new space



Example 1: Translate and scale digit images to fit in box of fixed size



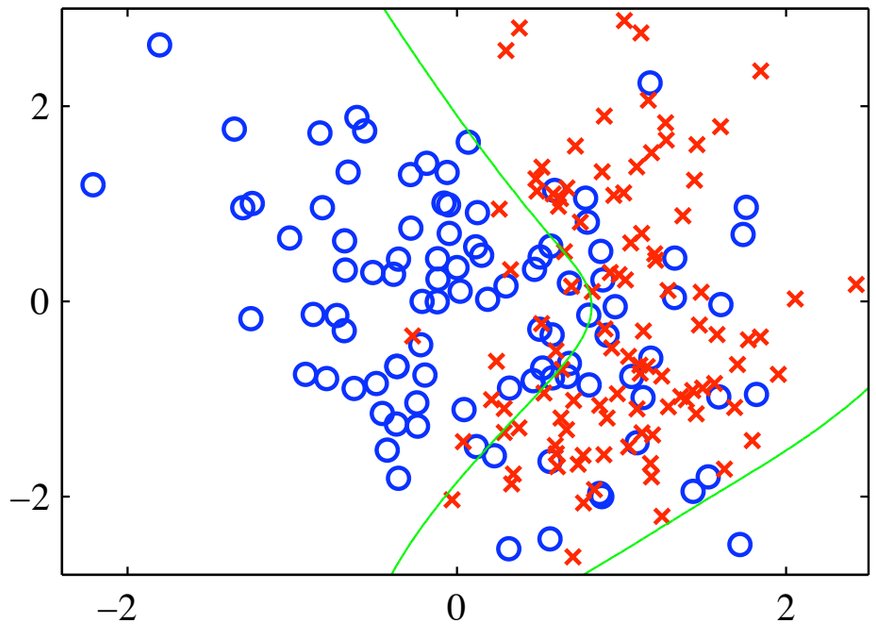
Example 2: Convert audio waveform to Fourier-based features

Kinds of Machine Learning

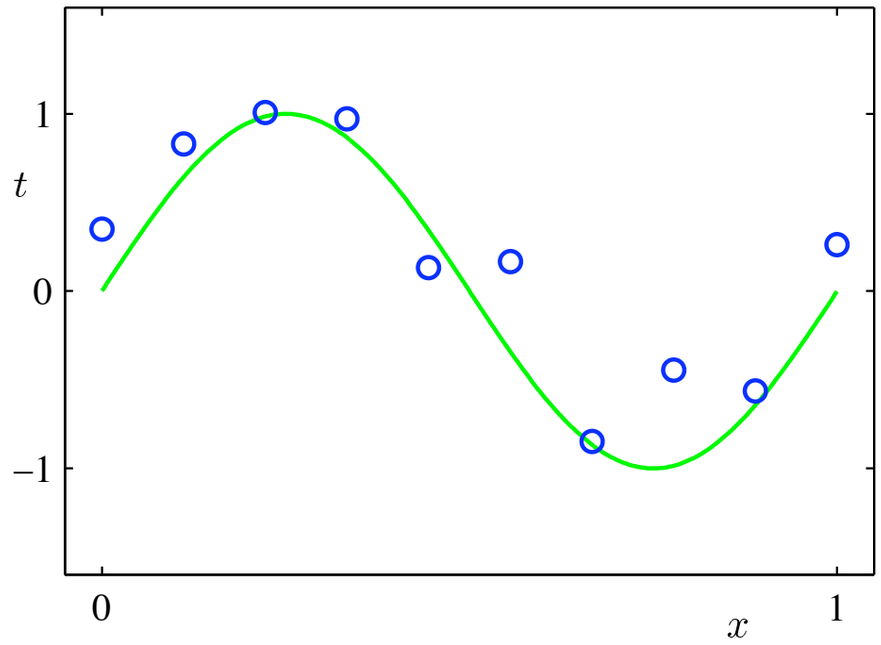
- *Supervised learning*:
  mapping inputs to targets
- *Unsupervised learning*:
  finding similar examples in data
- *Semi-supervised learning*:
  combining labeled and unlabeled examples
- *Reinforcement learning*:
  maximizing reward via appropriate action

Supervised Learning

Input vector $\mathbf{x}_i$ is matched to a target vector $\mathbf{t}_i$



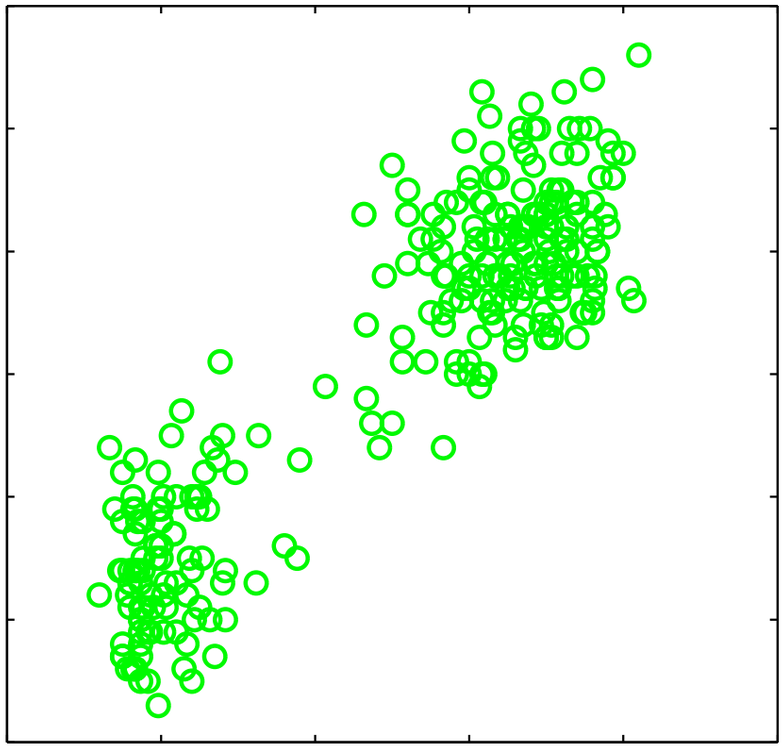
Classification:
$\mathbf{t}_i$ falls into discrete categories

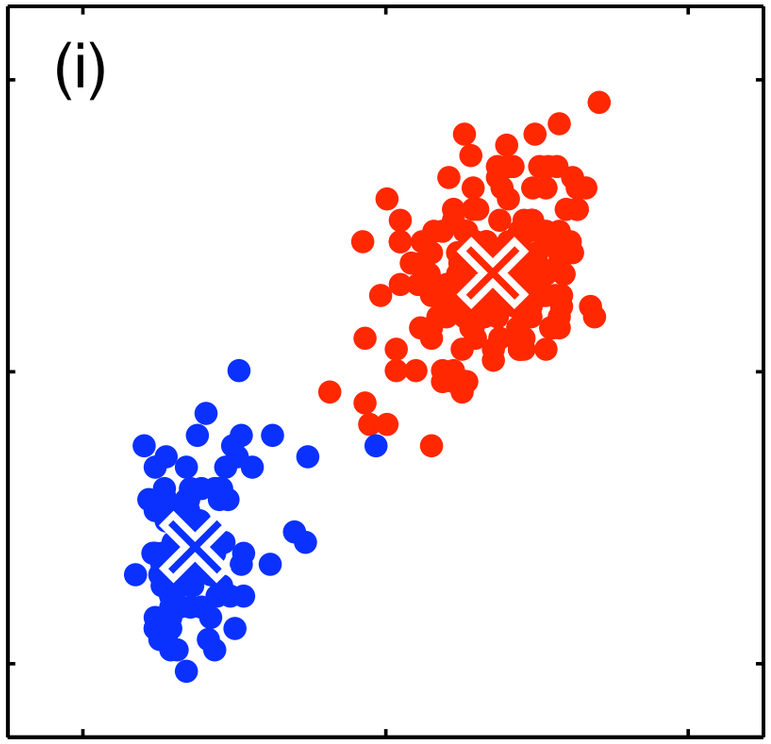


Regression :
$\mathbf{t}_i$ is continuous

Unsupervised Learning

Only input vector $\mathbf{x}_i$ present; no $\mathbf{t}_i$

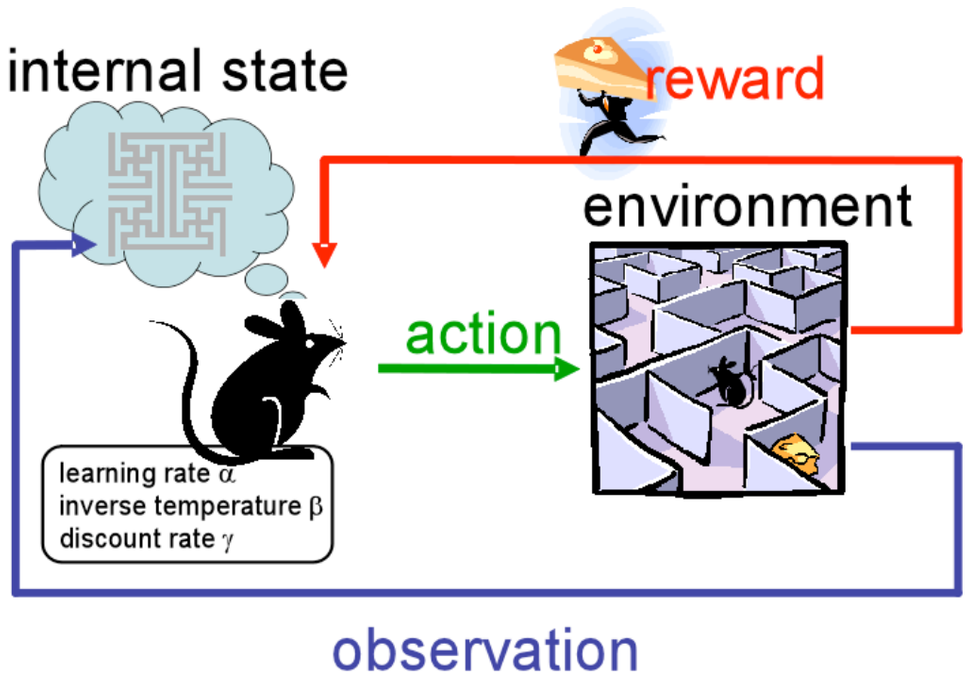




- Clustering (above): Discover groups of similar examples within data
- Density estimation: Determine distribution of data
- Dimensionality reduction: Find low-dimensional representations for, e.g., visualization

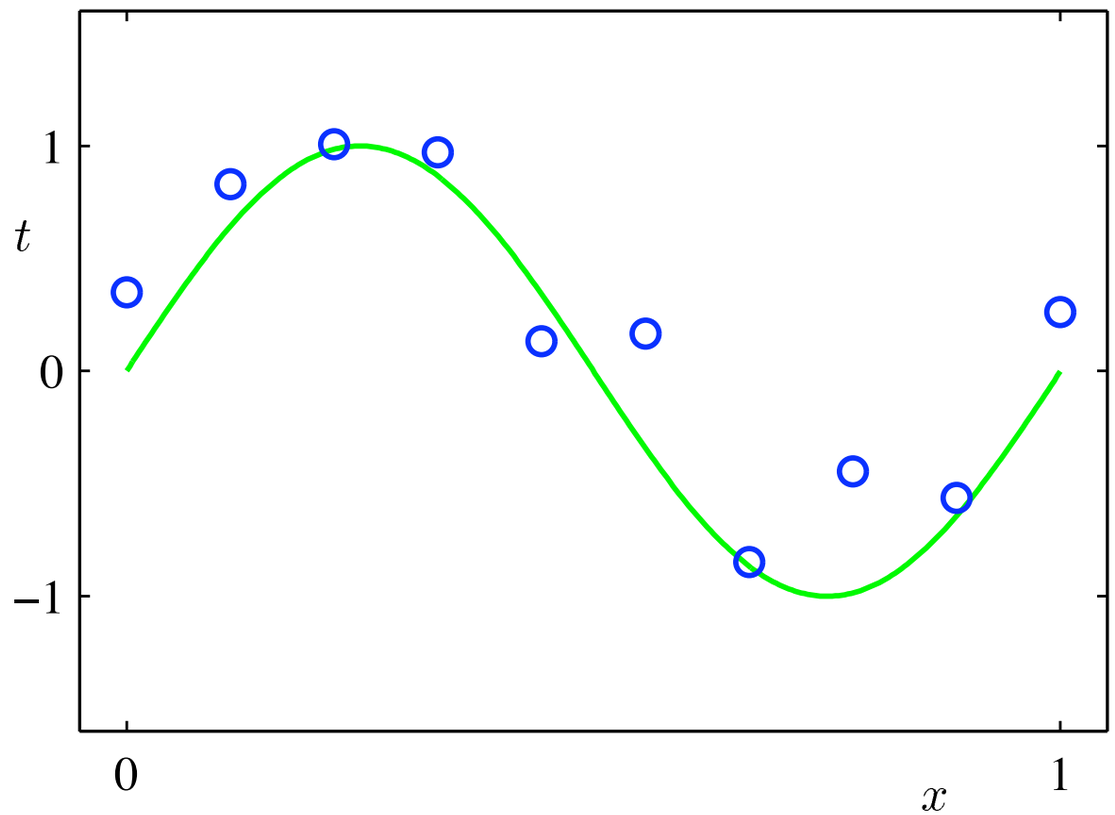
Reinforcement Learning

- Find suitable actions to take in a given situation in order to maximize reward
- No explicit training targets
- Discovery via trial and error



Polynomial curve fitting

- Training set: *N* observations of *x*; $X \equiv (x_1, ..., x_N)^T$
- Targets: *N* observations of *t*; $T \equiv (t_1, ..., t_N)^T$
- Goal is *generalization:* predict *x* for some unseen *t*



Plot of training data (10 points).
Green curve shows function sin(2 π x) used to generate data

Polynomial curve fitting

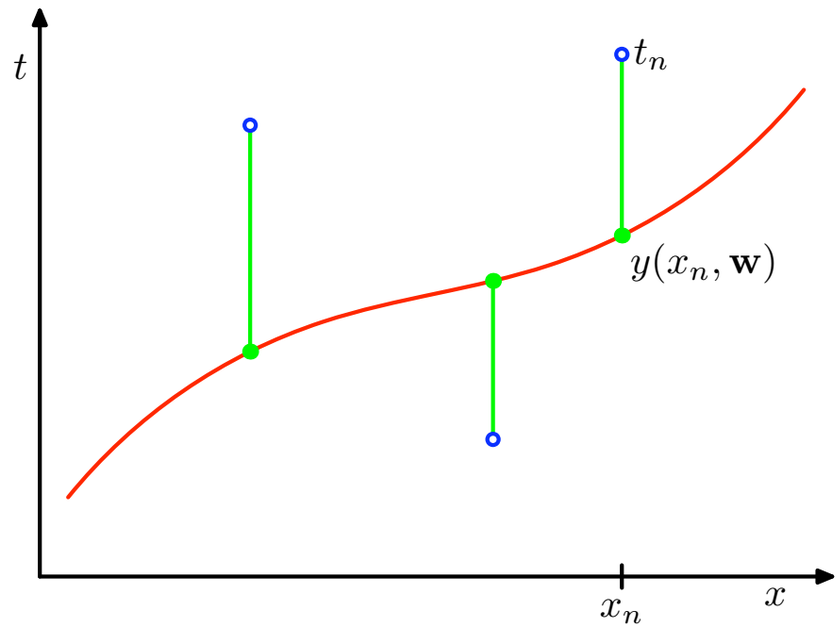
- Polynomial functional form:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- Fix coefficient values **w** via error minimization.
  Simple choice: minimize sum of the squares of the errors between predictions $y(x_n, \mathbf{w})$ for each point x_n and corresponding target values t_n

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(n_n, \mathbf{w}) - t_n)^2$$

Error function

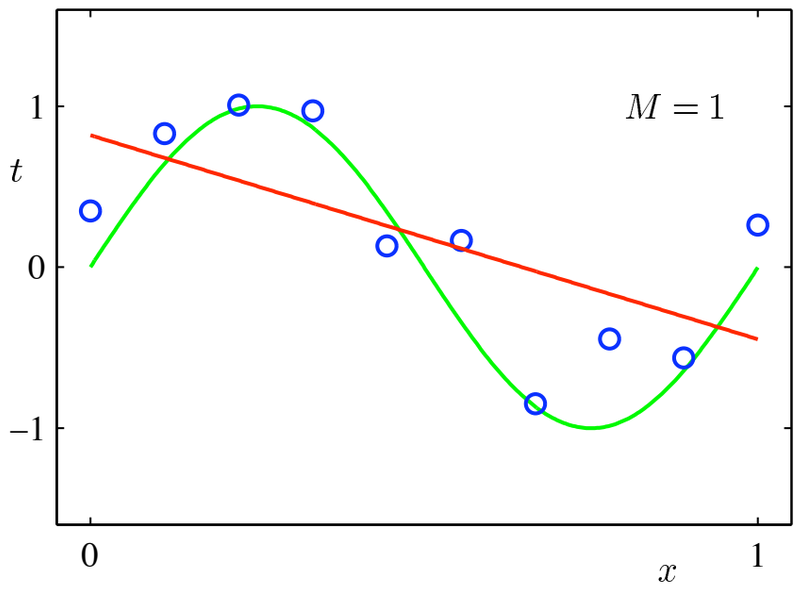


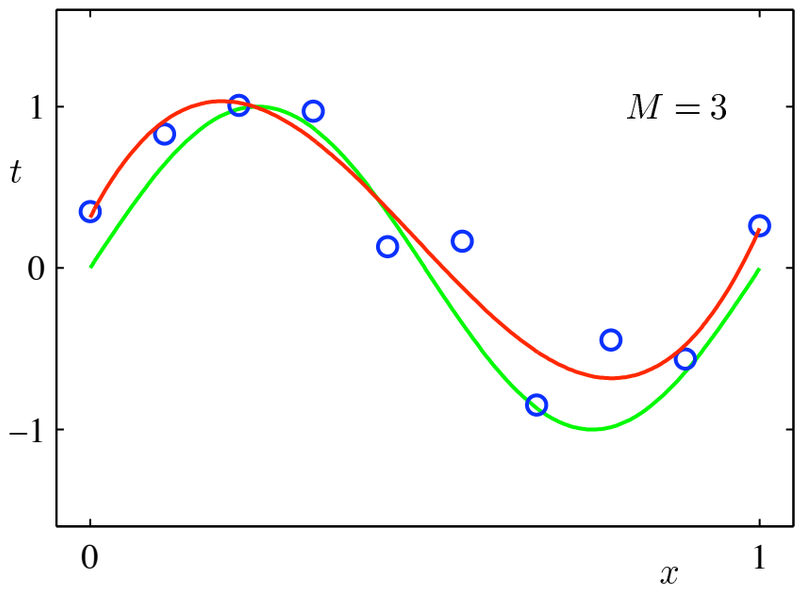
The error function corresponds to (one half of) the sum of the squares of the displacements (shown by vertical green bars) of each data point from the function y(x,**w**)

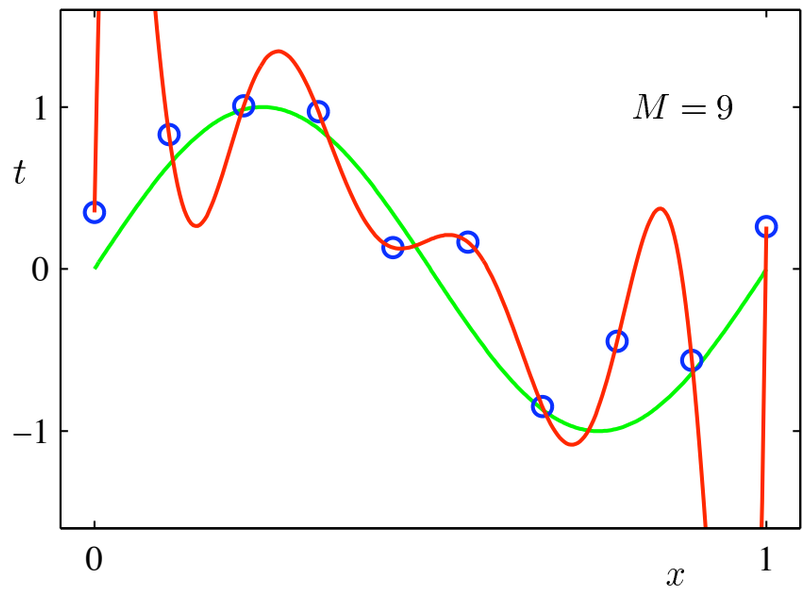
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(n_n, \mathbf{w}) - t_n)^2$$

- Error is quadratic function of coefficients **w**
- Minimization of function is thus unique

Model selection



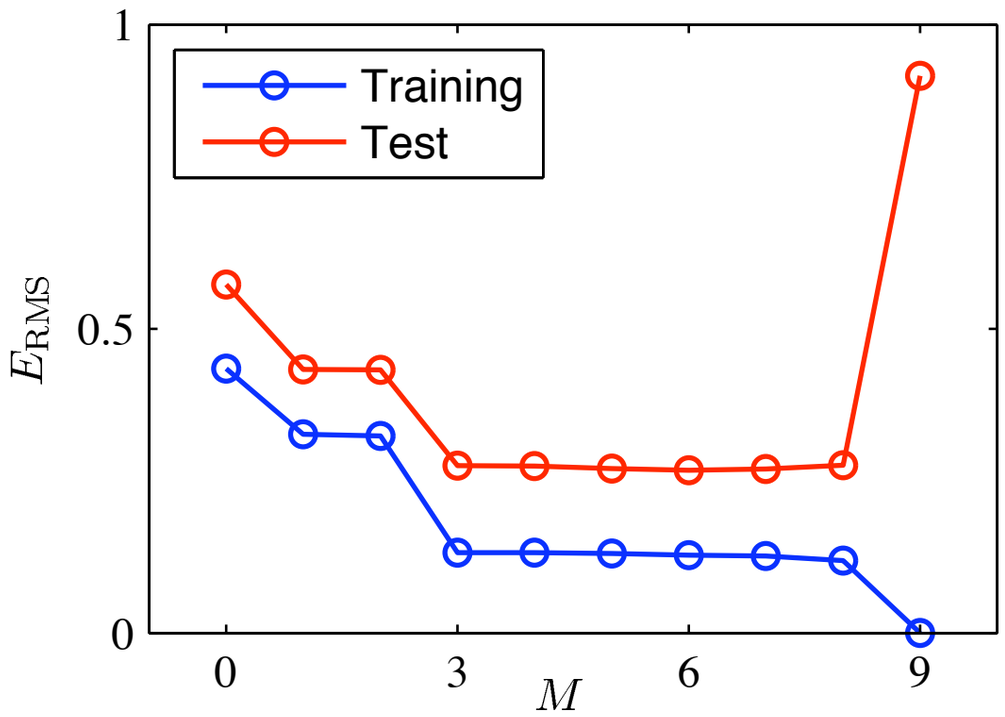




- What is the appropriate polynomial order?
- Balance over-fitting and under-fitting
- RMS allows comparison between different datasets

$$E_{RMS} = \sqrt{2E(\mathbf{w}*/\mathbf{N})}$$

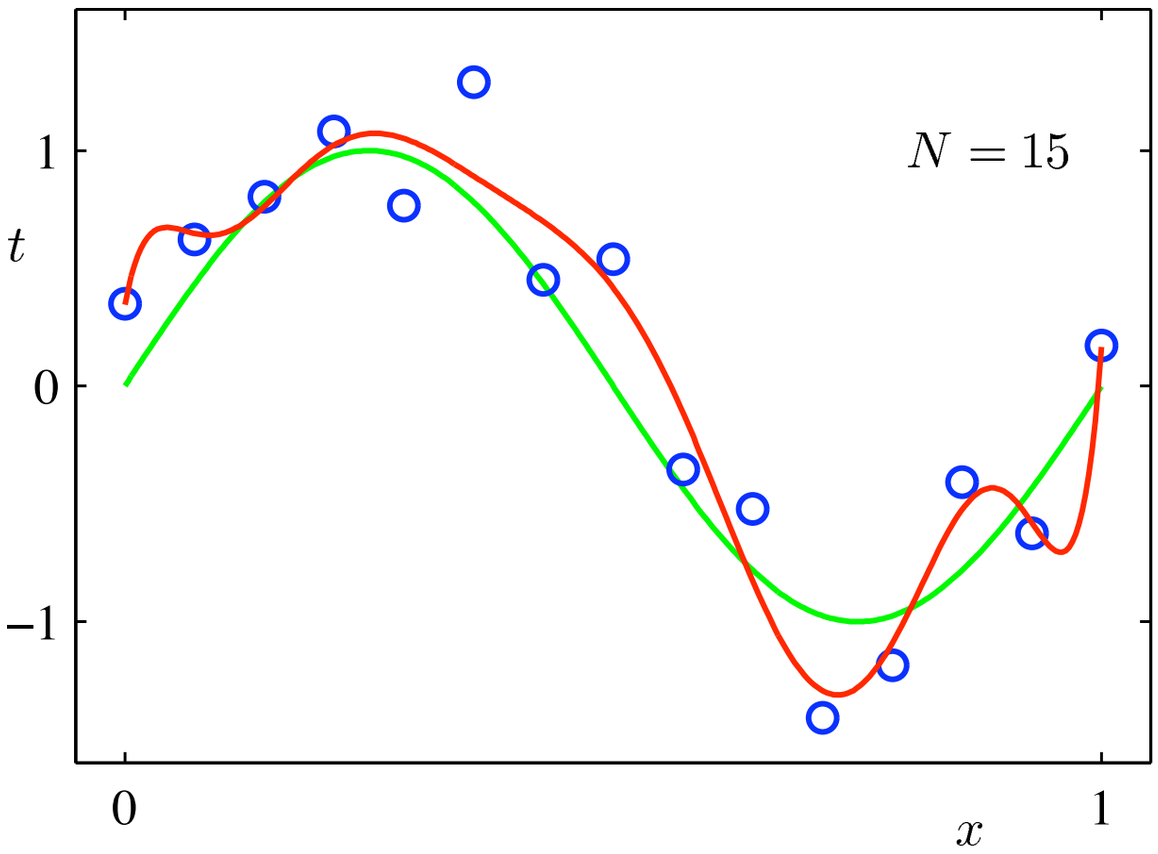
Training versus testing

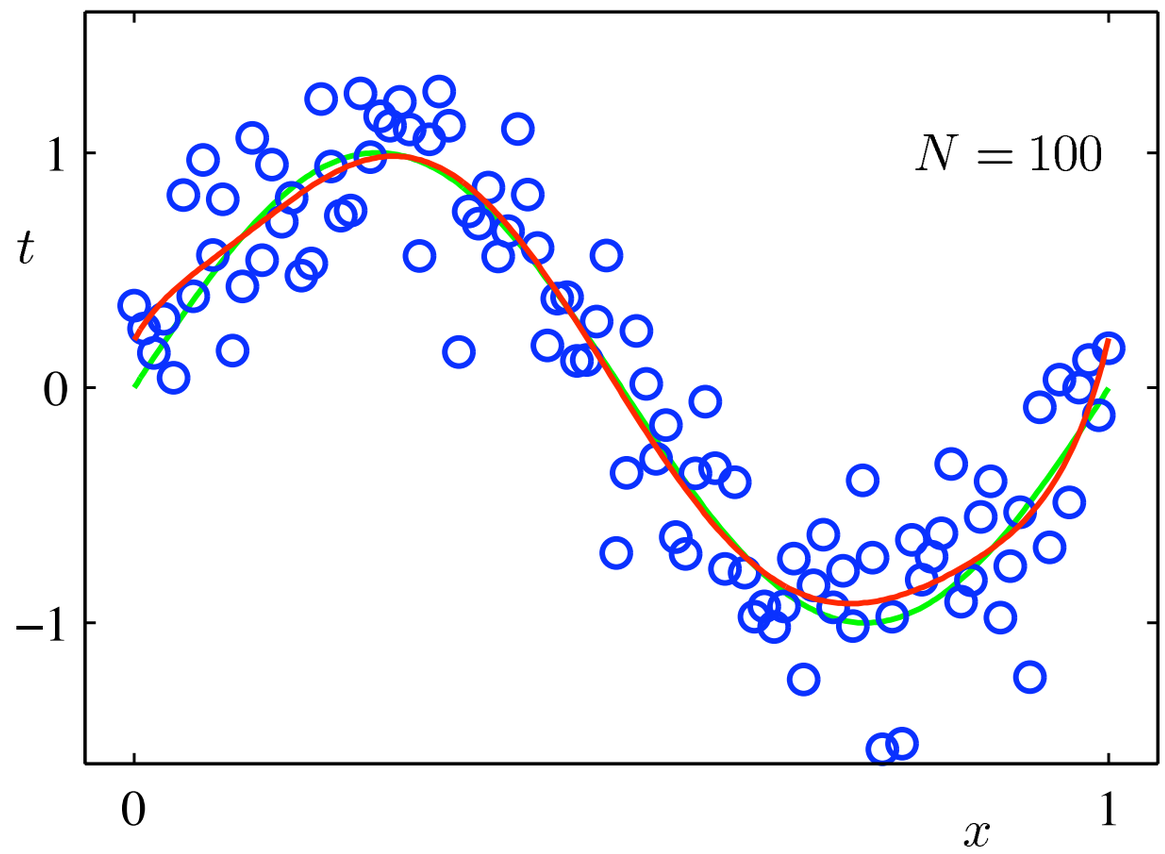


Graph of the RMS error evaluated on training set and on an independent test set.

- What is the best model given these results?
- Why does M=3 perform better than M=9? (Taylor expansion of generating function sine suggests that even M=∞ should work!)

Overfitting as function of training set size





Plots of the solutions obtained by minimizing using M=9 for N=15 data points (left) and N=100 data points (right).

Overtuning of parameters

- Magnitude of coefficients increases dramatically with model size
- M=9 can pass through all data points for N=10
- Minimizing least squares is example of *maximum likelihood*; Overfitting is general problem
- Many solutions; With Bayesian model, the *effective* number of parameters adjusts automatically to size of dataset.

	M=0	M=1	M=6	M=9
w^*_0	0.19	0.82	0.31	0.35
w^*_1		1.27	7.99	232.37
w^*_2			25.43	5321.83
w^*_3			17.37	48568.31
w^*_4				23163.30
w^*_5				640042.26
w^*_6				10618000.52
w^*_7				1042400.18
w^*_8				557682.99
w^*_9				125201.43

Table of coefficients for w* the unique solution of minimization of RMS for various polynomial orders.

Regularization

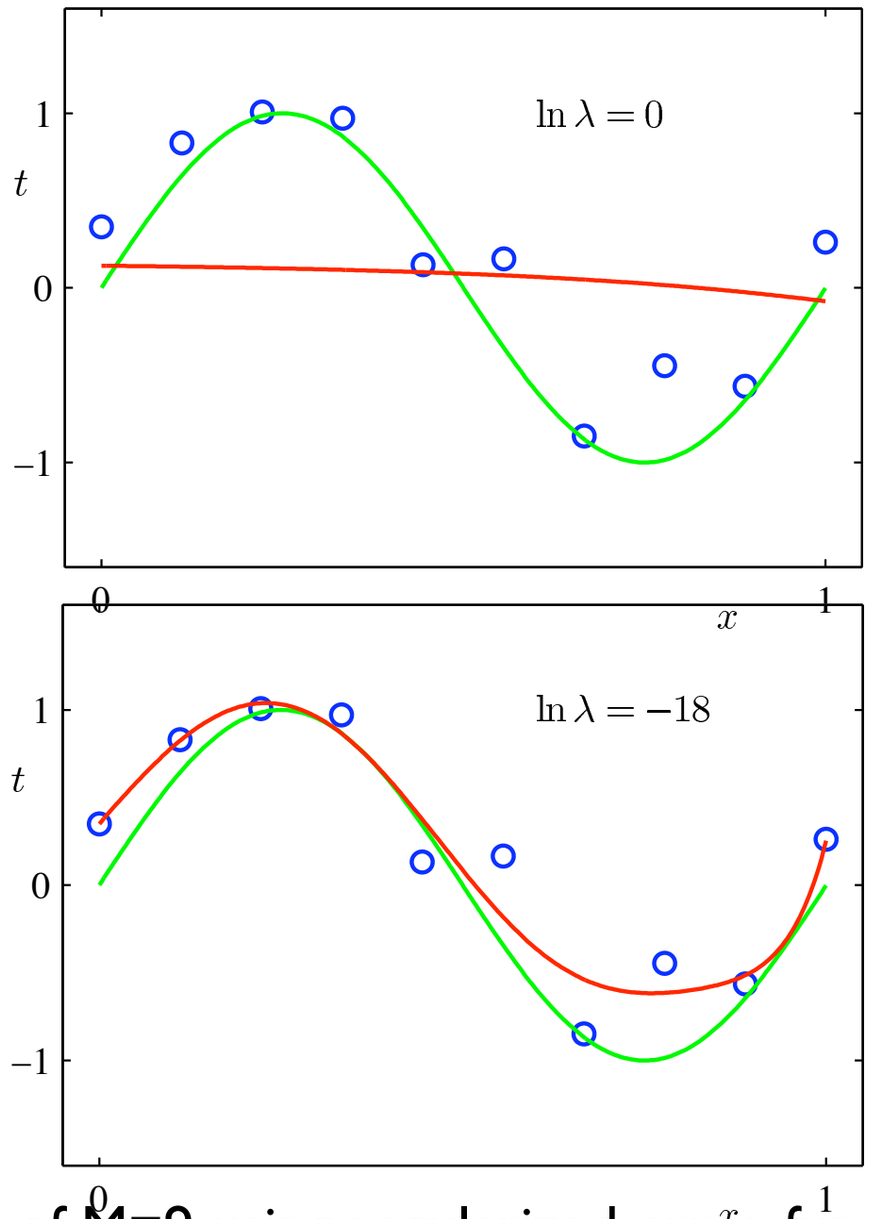
- Large weights generally lead to inflexible solutions
- Add penalty term to error function

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}(y(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

where

$$||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 + ... + w_M^2$$

- Statistics term: *shrinkage*
- Quadratic regularization yields *ridge regression*
- Neural networks : *weight decay*



Plots of M=9 using regularized error function on same 10-point dataset as before

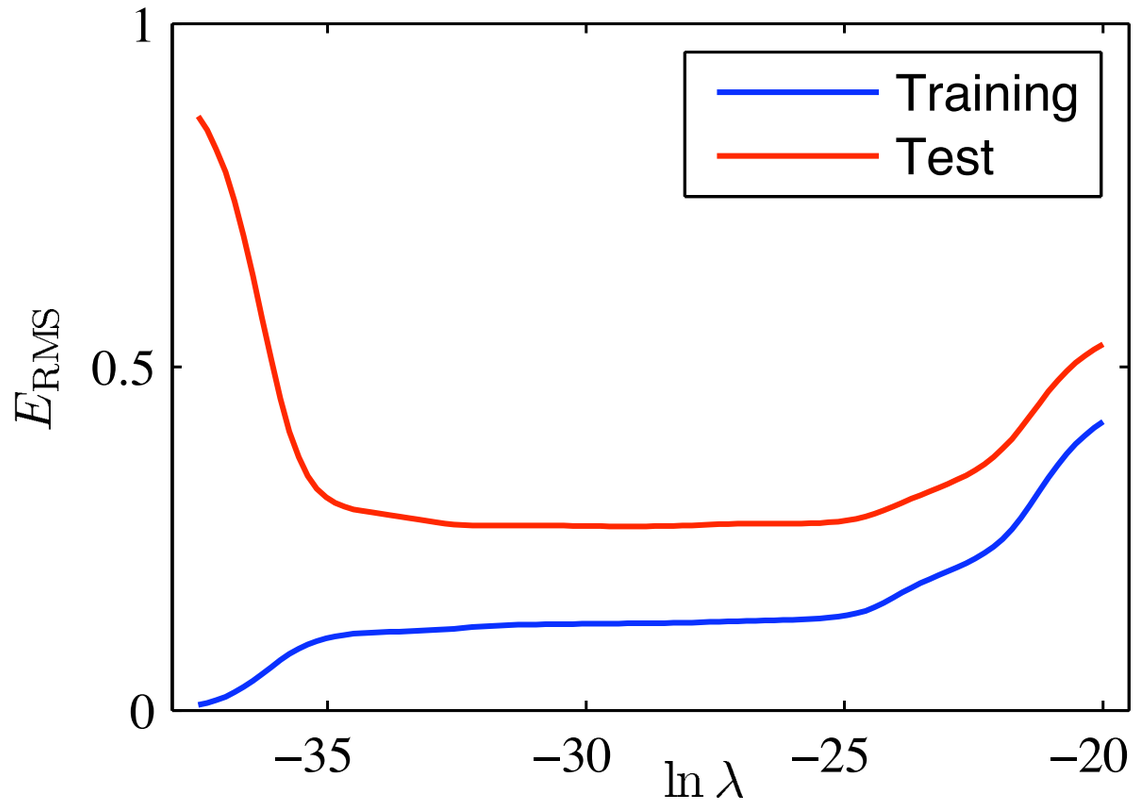
Results with regularization

	ln λ= -∞	ln λ= -18	ln λ= 0
w^*_0	0.35	0.35	0.13
w^*_1	232.37	4.74	0.05
w^*_2	5321.83	0.77	0.06
w^*_3	48568.31	31.97	0.05
w^*_4	23163.30	3.89	0.03
w^*_5	640042.26	55.28	0.02
w^*_6	10618000.52	41.32	0.01
w^*_7	1042400.18	45.95	0.00
w^*_8	557682.99	91.53	0.00
w^*_9	125201.43	72.68	0.10

Table of coefficients for w* the unique solution of minimization of RMS for various values for regularization parameter **λ** in

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}(y(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

Results with regularization



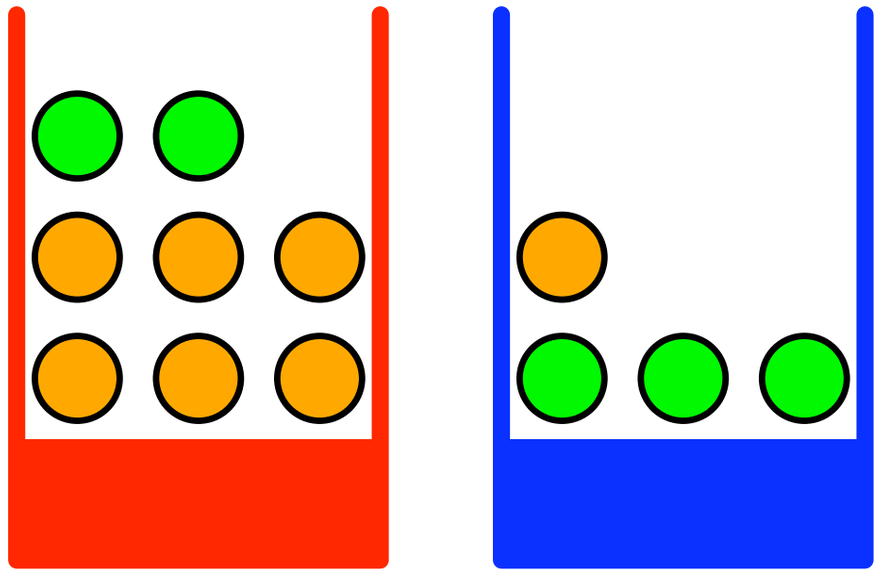
RMS error versus ln λ for the M=9 polynomial

- Regularization effectively controls complexity of model
- Regularization parameter (λ) a *hyperparameter* of model.
- Possible to overfit hyperparameters
- Simple safeguard: use *validation set* (distinct from *test set* and *training set*) to optimize model complexity

Generalization is difficult

- In principle, any number of consistent solutions may exist
- Occam's Razor: prefer the simplest solution. But what is "simple"?
- With curve fitting, perhaps smooth == simple
- Can use prior knowledge to *rank* solutions
  E.g. prefer a sparse model for regularization and efficiency
- ML searches in the space of possible *models;*
  Models themselves search through hyperparameters and parameters
- ML balances engineering, embedding prior knowledge in the model, cost of obtaining data, etc. . .

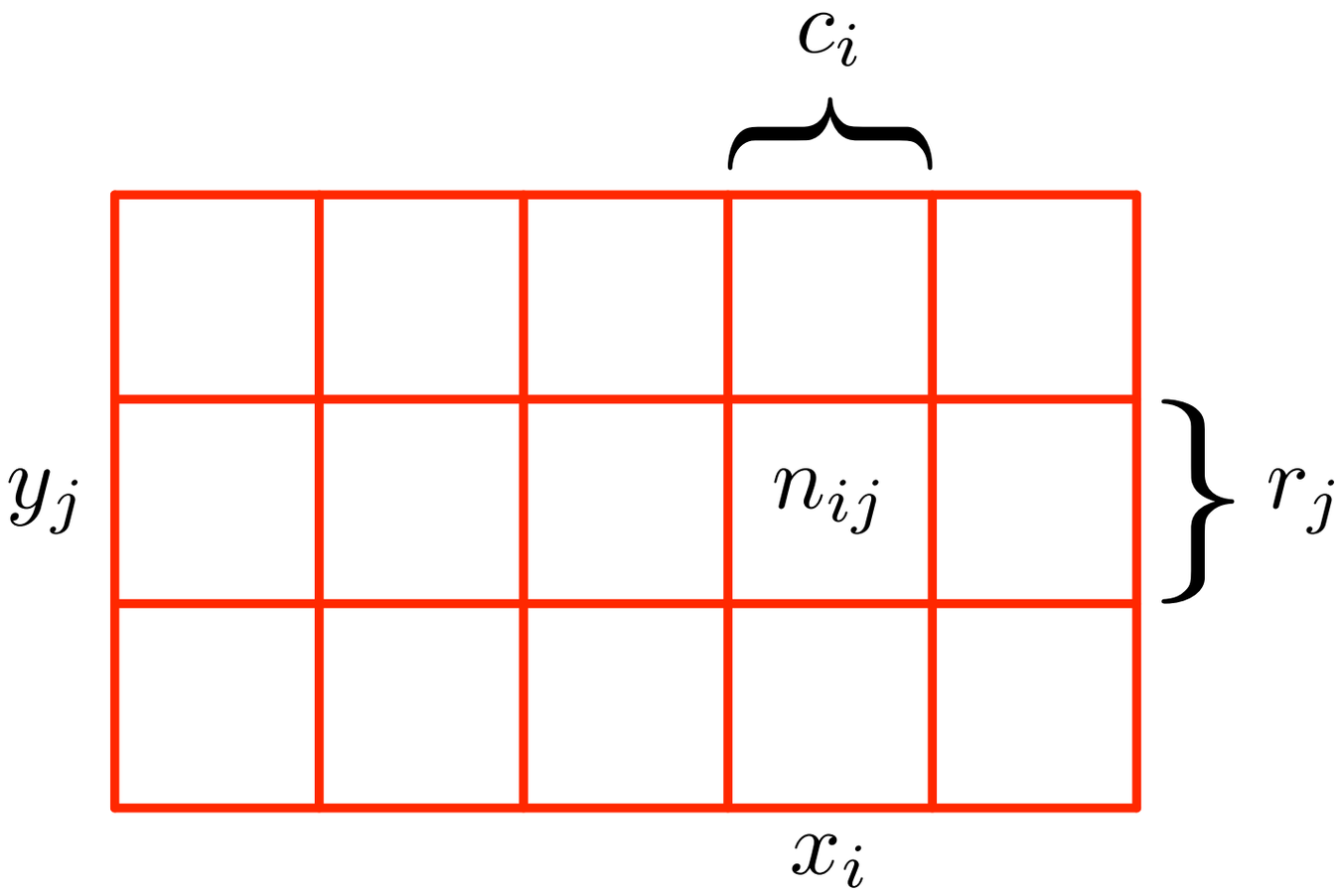
Probability Theory



Simple example of two coloured boxes each containing frut (apples in green, oranges in orange)

- Randomly select a box *B* = *r* or *b* [red or blue] such that p(B = r) = .6
- Then randomly select a piece of fruit *F* = *a* or *o* [apple or orange] with equal probability across pieces of fruit in a box

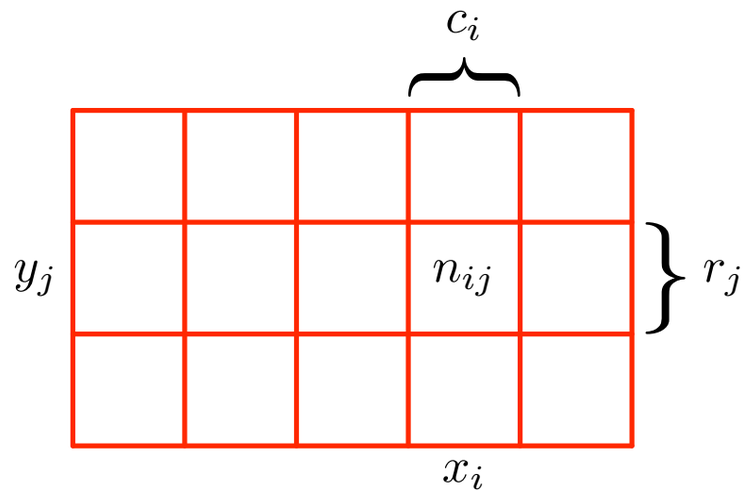
Probability by counting



Example for deriving sum and product rules using 2 random variables *X* and *Y*

- 2 random variables *X* and *Y*;
- *N* trials;
- *X* takes values x_i ... x_M;
- *Y* takes values y_j ... y_L;
- Number where $X=x_i$ and $Y=y_j$ is n_{ij}
- Number where $X=x_i$ is c_i
- Number where $Y=y_j$ is r_i

Sum and Product Rules



Joint probability: $p(X = x_i, Y = y_j) = \dfrac{n_{ij}}{N}$

Marginal probability: $p(X = x_i) = \dfrac{c_i}{N}$ $\qquad (c_i = \sum_j n_{ij})$

Sum rule: $p(X = x_i) = \sum_{j=1}^{L} p(X = x_i, Y = y_j)$

Conditional probability: $p(Y = y_j | X = x_i) = \dfrac{n_{ij}}{c_i}$

Product rule: $p(X = x_i, Y = y_j) = \dfrac{n_{ij}}{c_i} \cdot \dfrac{c_i}{N}$

$= p(Y = y_j | X = x_i) p(X = x_i)$

Bayes' Theorem

Sum rule:

$$p(X = x_i) = \sum_{j=1}^{L} p(X = x_i, Y = y_j)$$

$$p(X) = \sum_{Y} p(X, Y)$$

Product rule:

$$p(X = x_i, Y = y_j) = p(Y = y_j | X = x_i) p(X = x_i)$$

$$p(X, Y) = p(Y|X)p(X)$$

Bayes' Theorem:

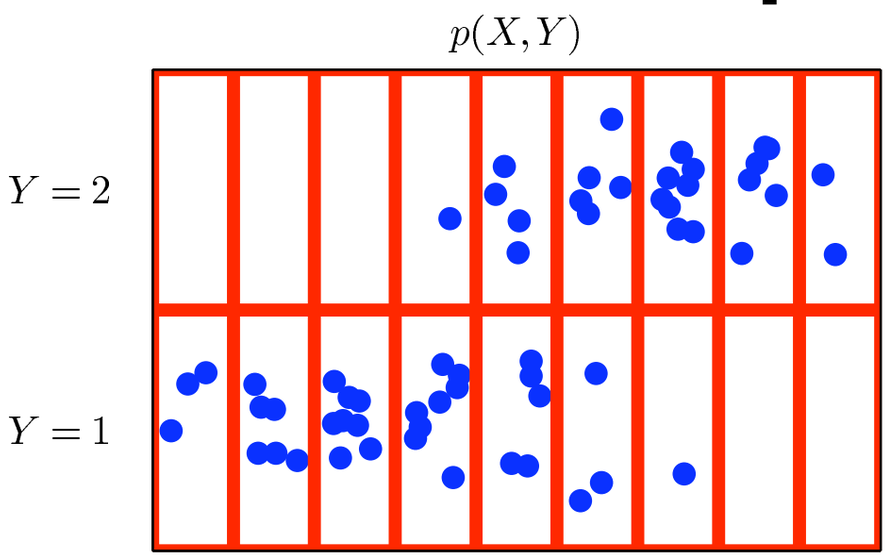
(From product rule plus symmetry property p(X,Y) = p(Y,X))

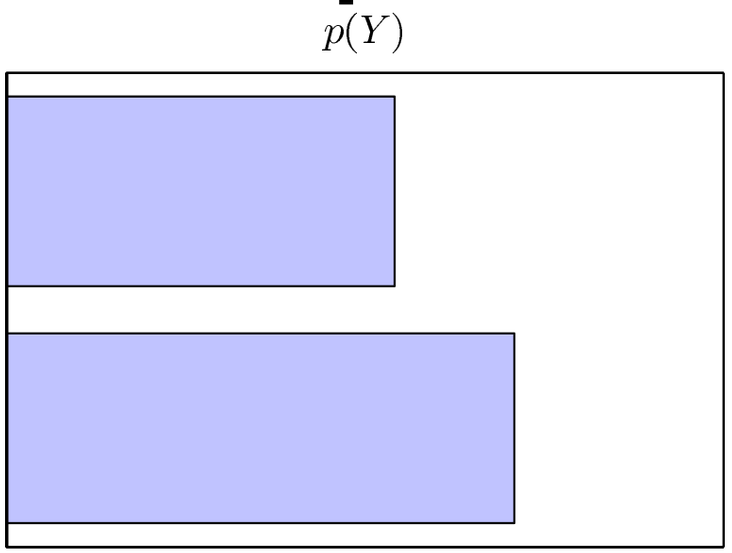
$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

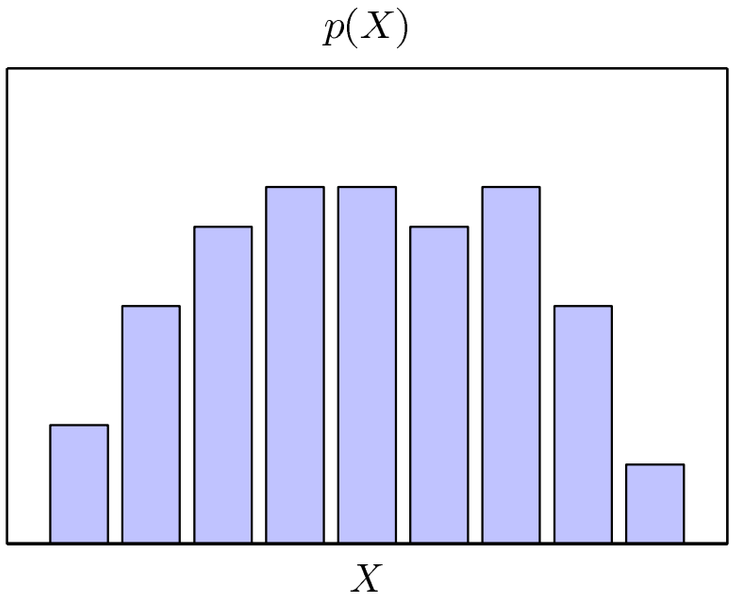
Denominator can be seen as normalizer to ensure conditionals sum to 1.0

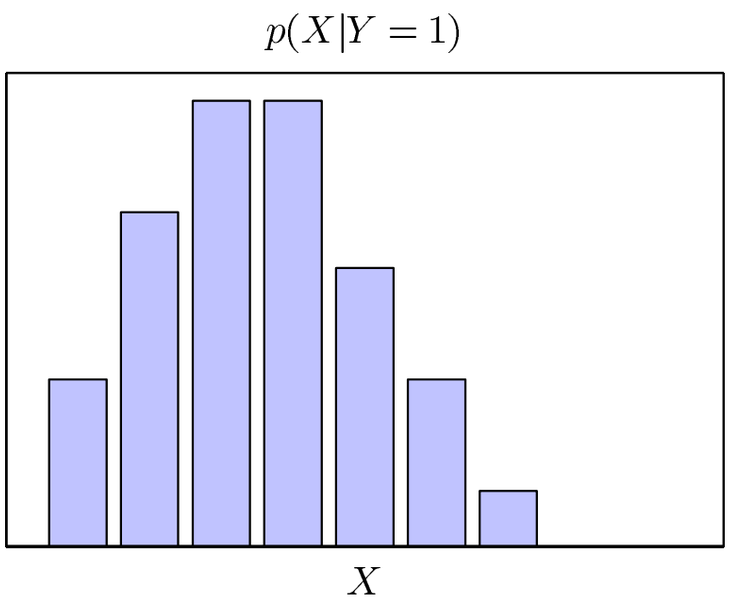
$$p(X) = \sum_{Y} p(X|Y)p(Y)$$

Simple example



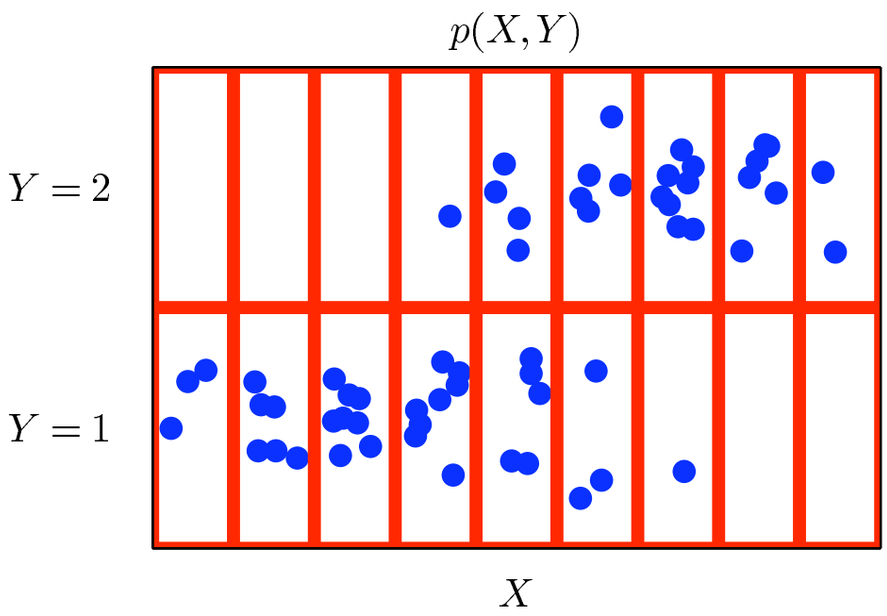


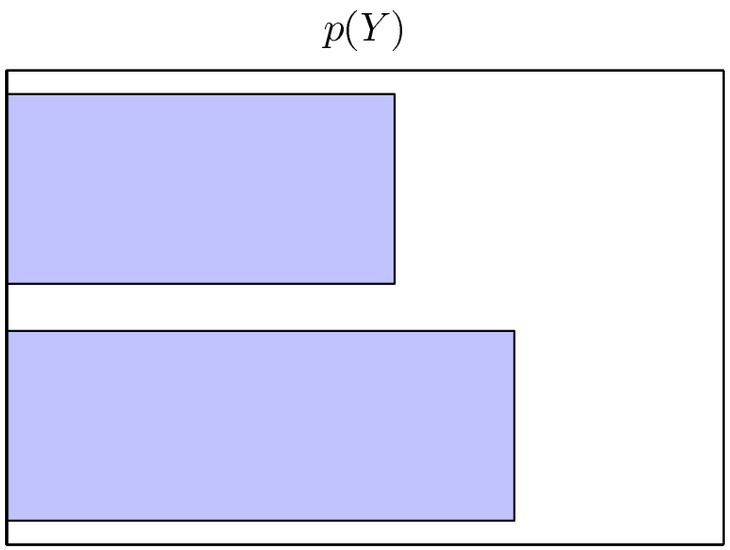




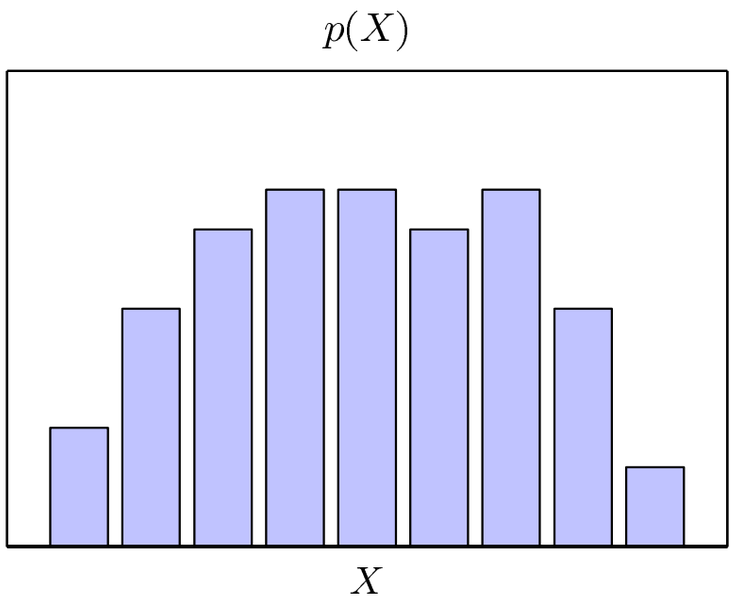
$$
\begin{aligned}
p(F=a) &= p(F=a|B=r)p(B=r) + p(F=a|b=b)p(B=b) \\
&= \frac{1}{4} \times \frac{4}{10} \times \frac{3}{4} \times \frac{6}{10} = \frac{11}{20}
\end{aligned}
$$

Marginal vs Conditional Distributions

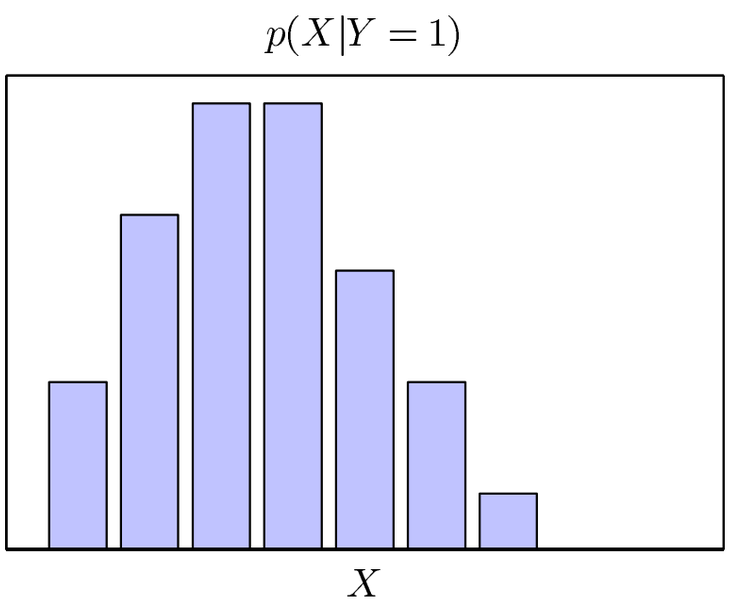




Marginal *p(Y)*

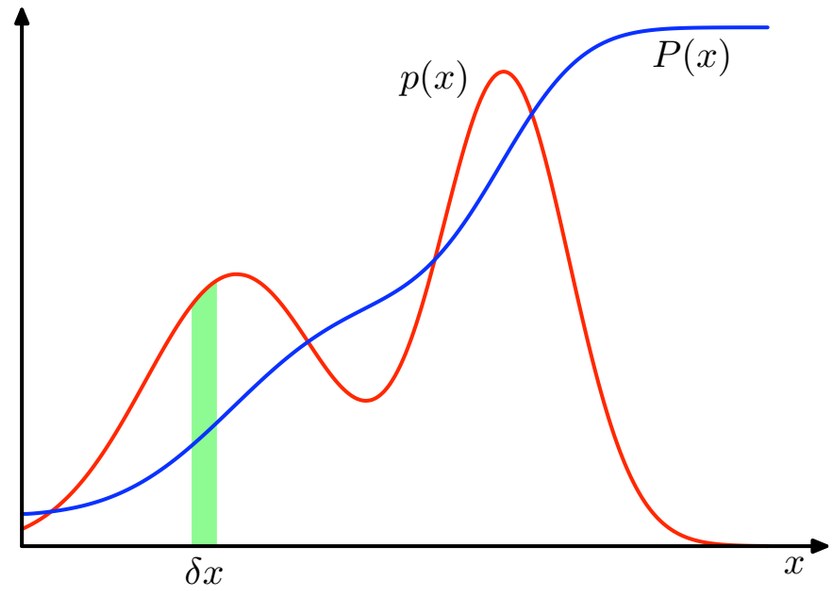


Marginal *p(X)*



Conditional *p(X|Y=1)*

Probability densities



Probability density *p(x)* over continuous variable *x*

- Probabilities over events can be extended to continuous variables
- Pr. of falling in interval (*x*, *x* + δ*x*) given by *p(x)*δ*x* for δ*x*→0
- Pr. that *x* will lie in interval (*a*,*b*) given by $p(x \in (a,b)) = \int_a^b p(x)\mathrm{d}x$
- Sum rule for densities: $p(x) = \int p(x,y)\mathrm{d}y$
- Product rule for densities: $p(x,y) = p(y|x)p(x)$

Expectations and covariances

Expectation of *f(x)* is average value of *f(x)* under prob. dist. *p(x)*

Discrete distribution: $$\mathbb{E}[f] = \sum_x p(x)f(x)$$

Continuous variables: $$\mathbb{E}[f] = \int p(x)f(x)\mathrm{d}x$$

Sample of *N* points: $$\mathbb{E}[f] \simeq \frac{1}{N}\sum_{n=1}^{N} f(x_n)$$

Conditional expectation: $$\mathbb{E}[f|y] = \sum_x p(x|y)f(x)$$

Variance and covariance of *f(x)*

Variance provides a measure of how much variability there is in the function *f(x)* and is defined in terms of expectation. Note that the variance of a variable *x* can be treated as a special case.

$$\begin{aligned}\mathrm{var}[f] &= \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] \\ \mathrm{var}[f] &= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)])^2 \\ \mathrm{var}[x] &= \mathbb{E}[x^2] - \mathbb{E}[x]^2\end{aligned}$$

Covariance expresses extent to which two variables *x* and *y* vary together.

$$\mathrm{cov}[x, y] = \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]$$

For vectors of random variables, covariance is a matrix.

$$\mathrm{cov}[\mathbf{x}, \mathbf{y}] = \mathbb{E}_{\mathbf{x},\mathbf{y}}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T]$$

Bayesian probabilities

- Make inference about the properties of our parameters **w** using p(**w**), called the *prior*
- Effect of observed data $D = \{t_1, t_2, \ldots t_N\}$ is expressed through conditional $p(D|\mathbf{w})$ and is called *likelihood*
- Evaluate uncertainty in **w** *after* observing D in form $p(\mathbf{w}|D)$, called the *posterior*

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

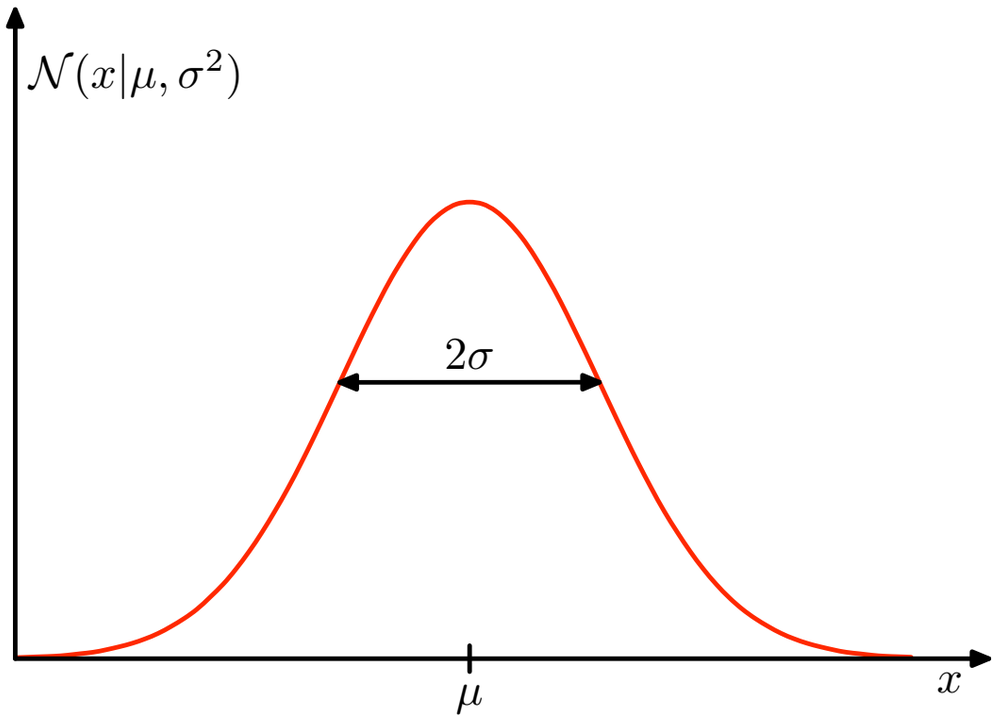
$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

- Denominator $p(D)$ is normalization term

Frequentist vs Bayesian

- Frequentist:
  - Set **w** to value such that $p(D|\mathbf{w})$ is maximized
  - **w** considered a fixed parameter
  - Use estimator, e.g., minimization of *negative log likelihood*
  - Consider distribution of data sets *D* (e.g. training /testing)
- Bayesian:
  - Only consider a single dataset *D*
  - Model uncertainty using distribution over **w**

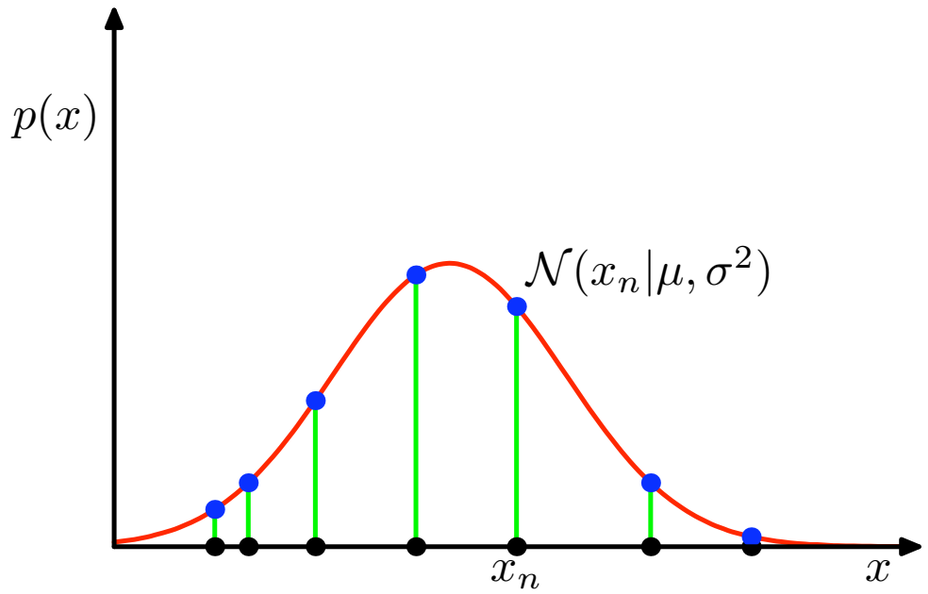
Gaussian distribution



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp\Big\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \Big\}$$

- Always positive; sums to 1.0
- precision $\beta = 1/\sigma^2$
- expectation $\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu,\sigma^2) x \, \mathrm{d}x = \mu$
- variance $\mathrm{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2$

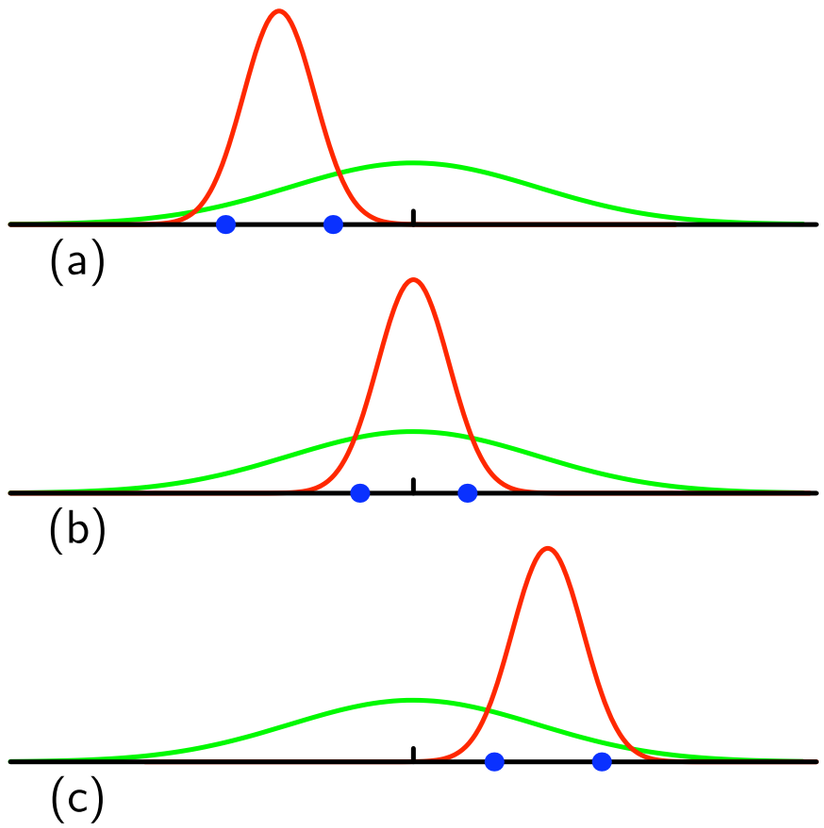
Maximizing likelihood



Likelihood function for Gaussian distribution (red). Black points denote data set $\{x_n\}$. Likelihood function corresponds to product of blue values.

- Maximum likelihood $$p(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu,\sigma^2)$$
- Log likelihood $$\ln p(\mathbf{x}|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\sigma)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$
- Sample mean via ML $$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n$$
- Sample variance via ML $$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{ML})^2$$

ML underestimates variance



Three samples from the green Gaussian, each consisting of 2 blue data points. Averaged across samples the mean is correct but the variance is systematically underestimated.

$$\mathbb{E}[\mu_{ML}] = \mu$$

$$\mathbb{E}[\sigma^2_{ML}] = \left(\frac{N-1}{N}\right)\sigma^2$$

$$\tilde{\sigma}^2 = \frac{N}{N-1}\sigma^2_{ML} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{ML})^2$$

Derivation of unbiased variance

Specific proof

We will demonstrate why s^2 is an unbiased estimator of the population variance. An estimator $\hat{\theta}$ for a parameter θ is unbiased if $\mathrm{E}(\hat{\theta}) = \theta$. Therefore, to prove that s^2 is unbiased, we will show that $\mathrm{E}(s^2) = \sigma^2$. As an assumption, the population which the x_i are drawn from has mean μ and variance σ².

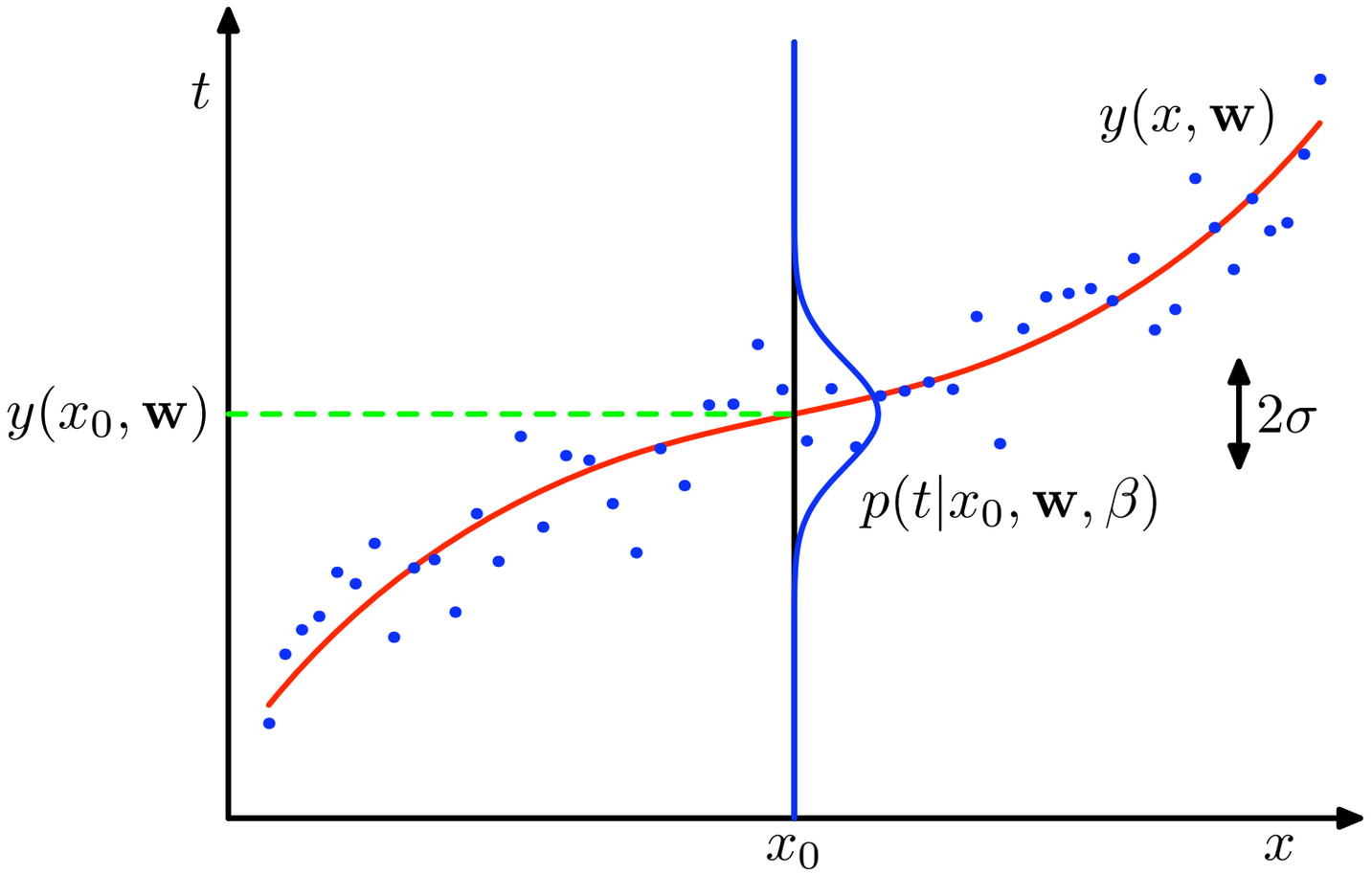
$$
\begin{aligned}
\mathrm{E}(s^2) &= \mathrm{E}\left(\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2\right) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\mathrm{E}\left((x_i-\overline{x})^2\right) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\mathrm{E}\left(((x_i-\mu)-(\overline{x}-\mu))^2\right) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\left\{\mathrm{E}\left((x_i-\mu)^2\right)-2\mathrm{E}\left((x_i-\mu)(\overline{x}-\mu)\right)+\mathrm{E}\left((\overline{x}-\mu)^2\right)\right\} \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\left[\sigma^2-2\left(\frac{1}{n}\sum_{j=1}^{n}\mathrm{E}\left((x_i-\mu)(x_j-\mu)\right)\right)+\frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n}\mathrm{E}\left((x_j-\mu)(x_k-\mu)\right)\right] \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\left(\sigma^2-\frac{2\sigma^2}{n}+\frac{\sigma^2}{n}\right) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\frac{(n-1)\sigma^2}{n} \\
&= \frac{(n-1)\sigma^2}{n-1} = \sigma^2.
\end{aligned}
$$

(From wikipedia's variance entry)

Curve fitting revisited

- Goal: predict target *t* for new values of input *x* on basis of training inputs $\mathbf{x}=(x_1,...,x_N)^T$ and targets $\mathbf{t}=(t_1,...,t_N)^T$
- To express uncertainty over targets, assume that given *x*, target *t* has Gaussian with mean equal to *y(x,*w) of the polynomial curve, thus:

$$p(t|x,\mathbf{w},\beta) = \mathcal{N}\left(t|y(x,\mathbf{w}),\beta^{-1}\right)$$



Curve fitting revisited

- Train using maximum likelihood. Assume samples independently drawn from $p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right)$

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|y(x_n, \mathbf{w}, \beta^{-1}\right)$$

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)$$

- ML solution $\mathbf{w_{ML}}$ obtained by maximizing w.r.t. w. Last 2 terms fall away. Scaling via β does not alter maximum. Thus maximizing likelihood is the same as minimizing sum-of-squares error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(n_n, \mathbf{w}) - t_n)^2$$

Curve fitting revisited

- Can also use ML to determine precision (maximize w.r.t. β) :

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{ML}) - t_n\}^2$$

- Now that we have predictions for **w** and β we can make predictions about *x*. These are now based on *predictive distribution:*

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1}\right)$$

- Consider Gaussian prior over **w**

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{ -\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\}$$

Maximizing posterior

- Recall that posterior is proportional to prior and likelihood:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

- Take negative log of eqn above and combine with:

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)$$

and:

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2}\exp\left\{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\}$$

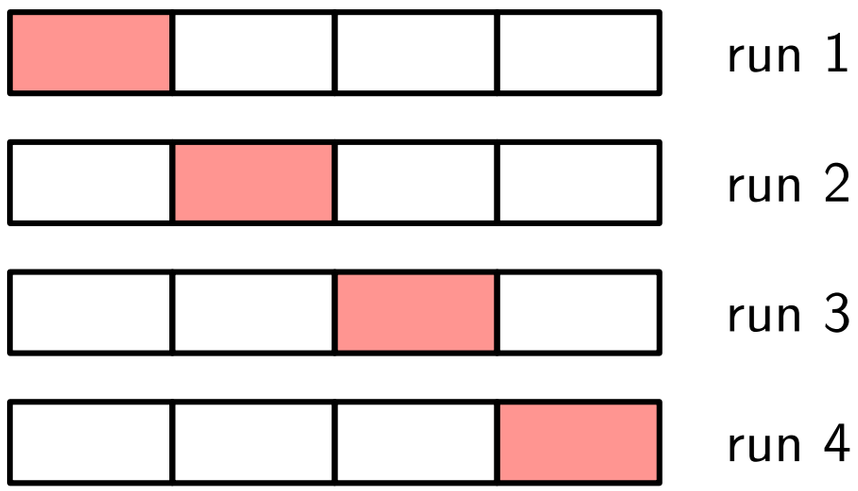
allows us to maximize posterior (MAP) via minimum of:

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$$

Yielding sum-of-squares with regularization of $\lambda = \alpha/\beta$

Model selection

- With ML, performance on training set not a good measure
- Can divide data into *training set* for fixing **w**, *validation set* for comparing models and *testing set* for final performance test
- With sparse data can re-use data using *cross-validation*



- However must train models multiple times; slow
- Many approaches exist (Bayesian versus non-Bayesian)
  Ex: Akaike information criterion (AIC) $\ln p(\mathcal{D}|\mathbf{w}_{\mathrm{ML}}) - M$
  which balances best-fit log likelihood with complexity of model (M = number of parameters). *[Later in the course...]*