

CS195-5 : Introduction to Machine Learning

Lecture 4

Greg Shakhnarovich

September 13 2006

Revised October 24th, 2006

Announcements

- Mailing list reminder
- Notes with derivations etc. on the website
- Problem set 1: out tonight, due Wed 9/27

Review

- Optimal regression function: $f^*(\mathbf{x}) = E_{p(y|\mathbf{x})} [y|\mathbf{x}]$
- Expected loss = structural error (limitations of function class)
+ approximation error (limitations of finite data)
- Statistical model for regression: $y = f(\mathbf{x}; \mathbf{w}) + \nu$
- Maximum likelihood estimation
 - For Gaussian noise, ML \equiv least squares.

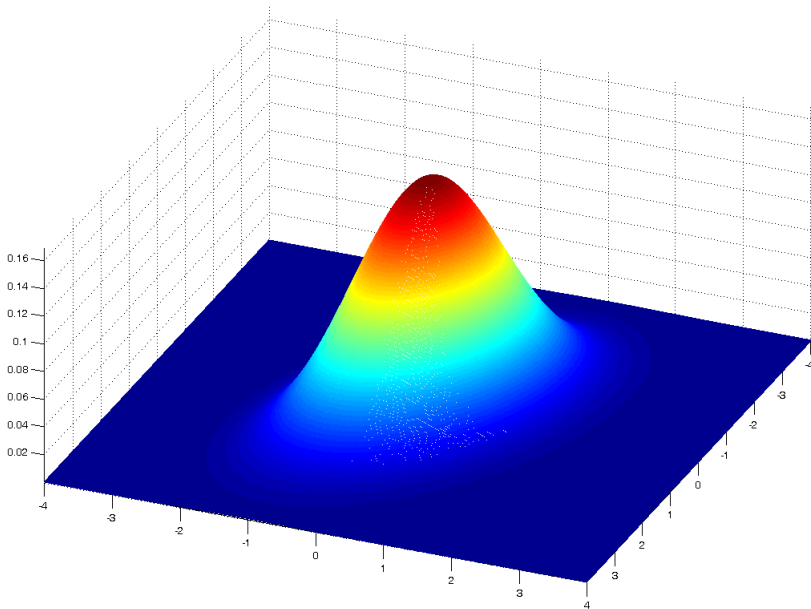
Today

- Review of multivariate Gaussian distributions
- Evaluating uncertainty in ML estimate of \mathbf{w}
- Extensions of linear regression
- Cross-validation
- Classification

Multivariate Gaussian distributions

- Gaussian distribution of a random vector \mathbf{x} in \mathbb{R}^d :

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

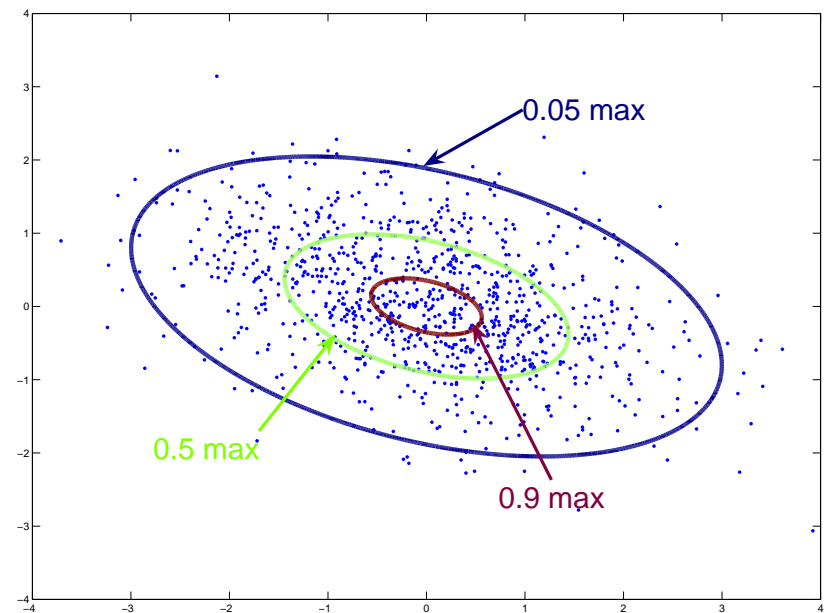


- The $\frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}}$ factor ensures it's a pdf (integrates to one).

Multivariate Gaussians: intuition

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

- This is the joint density of x_1, \dots, x_d .
- density falls off exponentially as a function of distance to the mean $\|\mathbf{x} - \mu\|$;
- the *covariance matrix* Σ determines the shape of the density;
- The higher d the faster $p(\mathbf{x})$ falls off.
- The determinant $|\Sigma|$ measures the “spread” (analogous to σ^2).



Multivariate Gaussians: properties

- Any linear function of a Gaussian is also a Gaussian: If $\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mu, \Sigma)$, and $\mathbf{z} = A\mathbf{x}$ then $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; A\mu, A\Sigma A^T)$.
- Any 1D conditional (a “slice” through a Gaussian) is also a Gaussian.
- Any marginal (distribution of a subset of dimensions) is also a Gaussian.
- A Gaussian pdf is completely determined by μ and Σ .
- Important case: “spherical”, or “isotropic” Gaussian, i.e. $\Sigma = \sigma^2 \mathbf{I}$
 - Components (dimensions) of \mathbf{x} are *uncorrelated*, and have the same 1D Gaussian distribution $\mathcal{N}(x_i; \mu_i, \sigma)$.

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- \mathbf{w}^* : the best linear regression function
- $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$: the ML estimate from training data.
- Assume the model is correct: $\mathbf{y} = \mathbf{X} \mathbf{w}^* + \nu$, with Gaussian noise $\nu \sim \mathcal{N}(\nu; \mathbf{0}, \sigma^2 \mathbf{I})$.
- If there were no noise ($\nu \equiv \mathbf{0}$):

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w}^*)$$

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- \mathbf{w}^* : the best linear regression function
- $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$: the ML estimate from training data.
- Assume the model is correct: $\mathbf{y} = \mathbf{X}\mathbf{w}^* + \nu$, with Gaussian noise $\nu \sim \mathcal{N}(\nu; \mathbf{0}, \sigma^2 \mathbf{I})$.
- If there were no noise ($\nu \equiv \mathbf{0}$):

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X}\mathbf{w}^*) \\ &= (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \mathbf{w}^* \\ &= \mathbf{w}^*\end{aligned}$$

i.e. the ML estimate would find the optimal \mathbf{w}^* .

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- When noise *is* present:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w}^* + \boldsymbol{\nu})\end{aligned}$$

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- When noise *is* present:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w}^* + \boldsymbol{\nu}) \\ &= (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \mathbf{w}^* + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{\nu}\end{aligned}$$

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- When noise *is* present:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w}^* + \nu) \\ &= (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \mathbf{w}^* + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \nu \\ &= \mathbf{w}^* \\ &\quad \text{correct parameters}\end{aligned}$$

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- When noise *is* present:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w}^* + \nu) \\ &= (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \mathbf{w}^* + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \nu \\ &= \underbrace{\mathbf{w}^*}_{\text{correct parameters}} + \underbrace{(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \nu}_{\text{estimate due to noise}}\end{aligned}$$

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

- When noise *is* present:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w}^* + \nu) \\ &= (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \mathbf{w}^* + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \nu \\ &= \underbrace{\mathbf{w}^*}_{\text{correct parameters}} + \underbrace{(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \nu}_{\text{estimate due to noise}}\end{aligned}$$

- What is the distribution of the random variable $\hat{\mathbf{w}}$ conditioned on the training data \mathbf{X} ?

Behavior of $\hat{\mathbf{w}}$ as an estimate of \mathbf{w}^*

$$\hat{\mathbf{w}} = \mathbf{w}^* + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{\nu}$$

- The noise is Gaussian $\sim \mathcal{N}(\boldsymbol{\nu}; \mathbf{0}, \sigma^2 \mathbf{I})$, $\hat{\mathbf{w}}$ given the data \mathbf{X} is a linear function of the noise.
 - Hence $\hat{\mathbf{w}}$ is also Gaussian.

- Its mean:

$$\mu_{\hat{\mathbf{w}}} = E[\hat{\mathbf{w}} | \mathbf{X}] = \mathbf{w}^*.$$

- Its covariance matrix:

$$\begin{aligned} \text{Cov}_{\hat{\mathbf{w}}} &= E\left[(\hat{\mathbf{w}} - \mu_{\hat{\mathbf{w}}})(\hat{\mathbf{w}} - \mu_{\hat{\mathbf{w}}})^T | \mathbf{X}\right] \\ &= \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}. \end{aligned}$$

Implications

- Confidence estimate on regression: *Assuming the model is correct*, we can report amount of uncertainty in our estimate $\hat{\mathbf{w}}$.

Implications

- Confidence estimate on regression: *Assuming the model is correct*, we can report amount of uncertainty in our estimate $\hat{\mathbf{w}}$.
- Active learning:
 - Suppose we have N examples, and can *select* the $N + 1$ -th one. I.e. we can ask an expert/“oracle” to label any example we want.
 - We should select an example that reduces our uncertainty most. Denote $\mathbf{X}' = \begin{bmatrix} \mathbf{X} \\ 1 & \mathbf{x}' \end{bmatrix}$. Then we should select

$$\mathbf{x}_{N+1} = \operatorname{argmin}_{\mathbf{x}'} |\operatorname{Cov}_{\hat{\mathbf{w}}}| = \operatorname{argmin}_{\mathbf{x}'} \left| (\mathbf{X}'^T \mathbf{X}')^{-1} \right|.$$

Example: polynomial regression in 1D

$$f(x; \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_mx^m.$$

- No longer linear in x – but still linear in \mathbf{w} !

Example: polynomial regression in 1D

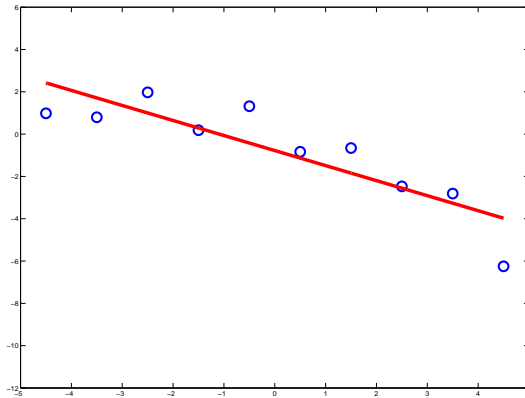
$$f(x; \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_mx^m.$$

- No longer linear in x – but still linear in \mathbf{w} !
- Define $\tilde{\mathbf{x}} = [1, x, x^2, \dots, x^m]^T$
- Then, $f(x; \mathbf{w}) = \mathbf{w}^T \tilde{\mathbf{x}}$ and we are back to the familiar simple linear regression.
The least squares solution:

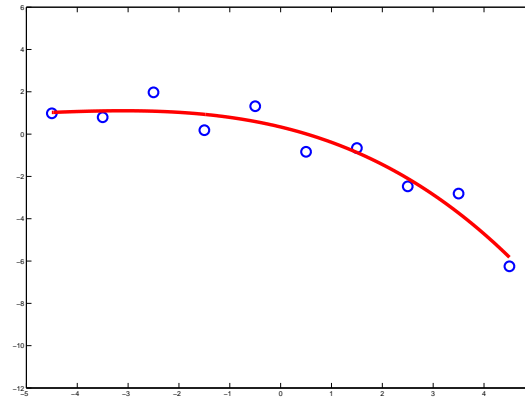
$$\hat{\mathbf{w}} = \left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \right)^{-1} \tilde{\mathbf{X}}^T \mathbf{y}, \quad \text{where} \quad \tilde{\mathbf{X}} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_N & x_N^2 & \dots & x_N^m \end{bmatrix}$$

Model complexity and overfitting

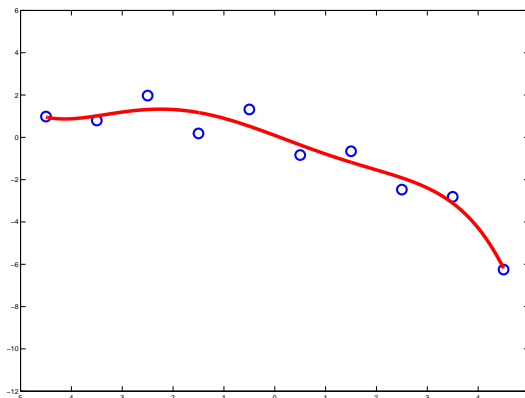
Data drawn from 3rd order model (`tryPolyFit.m`, `polyfitdata.mat`):



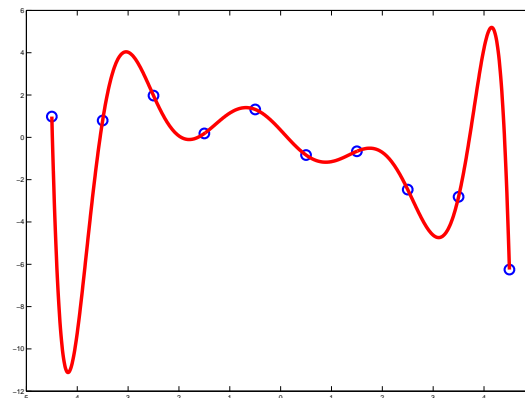
$m = 1$



$m = 3$



$m = 5$



$m = 10$

Cross-validation

- The basic idea: if a model overfits (is *too sensitive* to data) it will also be unstable. I.e. removal part of the data will change the fit significantly.
- We can *hold out* part of the data, fit the model to the rest, and then test on the heldout set.
- What are the problems of this approach?

Cross-validation

- The basic idea: if a model overfits (is *too sensitive* to data) it will also be unstable. I.e. removal part of the data will change the fit significantly.
- We can *hold out* part of the data, fit the model to the rest, and then test on the heldout set.
- What are the problems of this approach?
 - If the heldout set too small, we are susceptible to chance.
 - If it's too large, we get overly pessimistic (training on too little data).

Cross-validation

- The improved holdout method: *k*-fold *cross-validation*
 - Partition data into *k* roughly equal parts;
 - Train on all but *j*-th part, **test** on *j*-th part



Cross-validation

- The improved holdout method: *k*-fold *cross-validation*
 - Partition data into *k* roughly equal parts;
 - Train on all but *j*-th part, **test** on *j*-th part

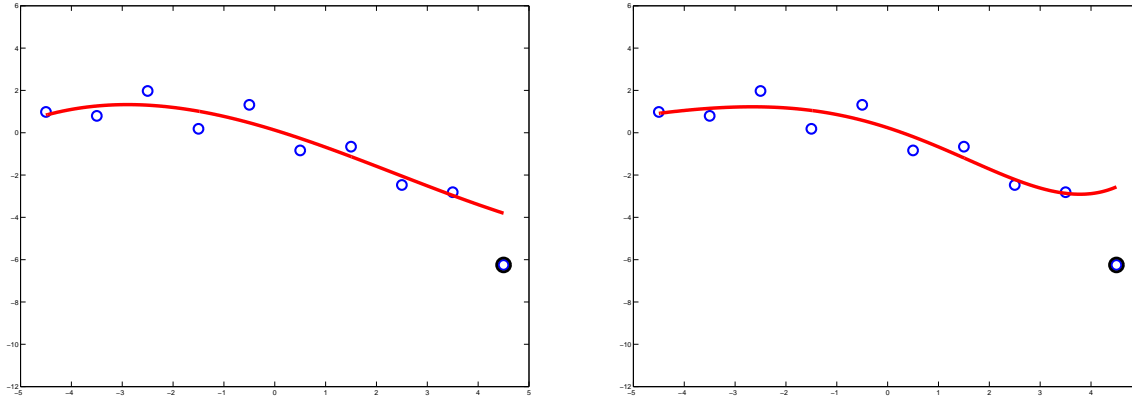


Cross-validation

- The improved holdout method: *k*-fold *cross-validation*
 - Partition data into *k* roughly equal parts;
 - Train on all but *j*-th part, **test** on *j*-th part

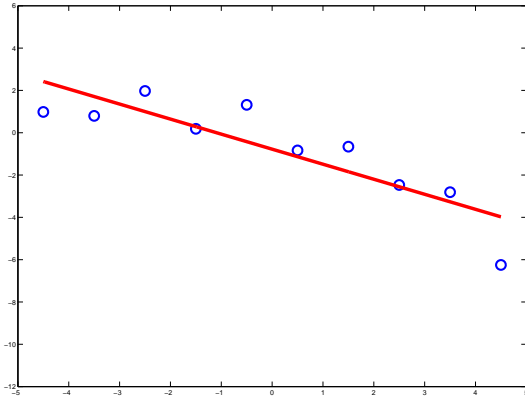


Cross-validation: example

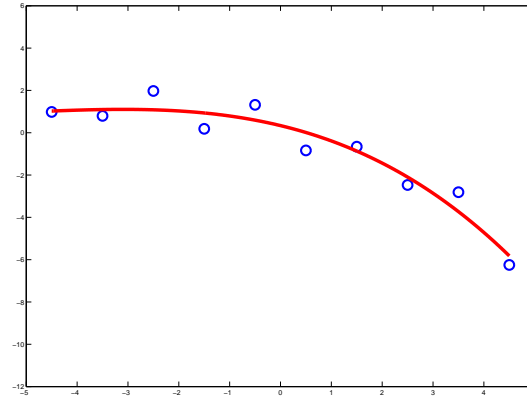


- This is a very good estimate, although expensive to compute
 - Need to run N estimation problems each on $N - 1$ examples!
 - An important research area: devising tricks for efficiently computing cross-validation estimates (by taking advantage of overlap between folds).

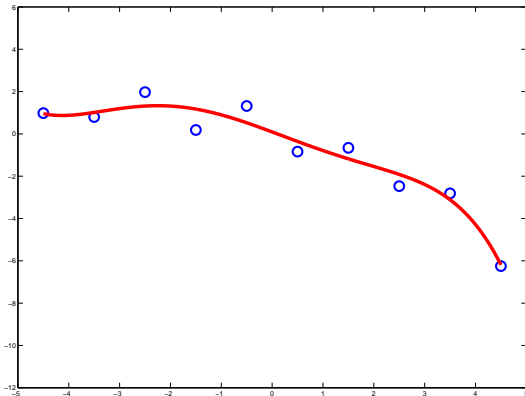
Cross-validation: example



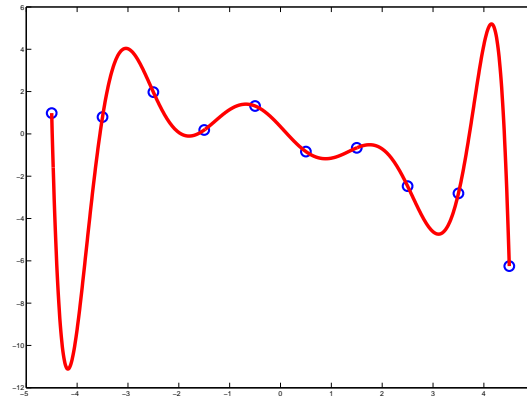
$$m = 1 : L_N = 1.4, \hat{L}_{CV} = 2.6$$



$$m = 3 : L_N = 0.4, \hat{L}_{CV} = 1.3$$



$$m = 5 : L_N = 0.3, \hat{L}_{CV} = 2.7$$



$$m = 10 : L_N = 0, \hat{L}_{CV} = 4 \times 10^4$$

General additive regression models

- A general extension of the linear regression model:

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \dots + w_m\phi_m(\mathbf{x}),$$

where $\phi_j(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}, j = 1, \dots, m$ are the *basis functions*.

- This is still linear in \mathbf{w} ;
- but may be non-linear in the inputs \mathbf{x} .

General additive regression models

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \dots + w_m\phi_m(\mathbf{x}),$$

- Still the same ML estimation technique applies:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

where \mathbf{X} is the *design matrix*

$$\begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_m(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_m(\mathbf{x}_2) \\ \dots & \dots & \dots & \dots & \dots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_m(\mathbf{x}_N) \end{bmatrix}$$

(for convenience we will denote $\phi_0(\mathbf{x}) \equiv 1$)

Examples of general linear regression

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \dots + w_m\phi_m(\mathbf{x}),$$

- In 1D, if $\phi_i(x) = x^i$, for $i = 0, \dots, m \Rightarrow$ we get the polynomial regression

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_mx^m.$$

- In \mathbb{R}^d , if $\phi_0(\mathbf{x}) = 1$, $\phi_i(\mathbf{x}) = x_i$ for $i = 1, \dots, d$ (projection on a single dimension) \Rightarrow we get simple linear regression:

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d.$$

Example: word frequency basis

- Let the space of \mathbf{x} be text documents (of varying sizes!)
- Let W_1, \dots, W_m be a set of m keywords.
- We can define a binary feature

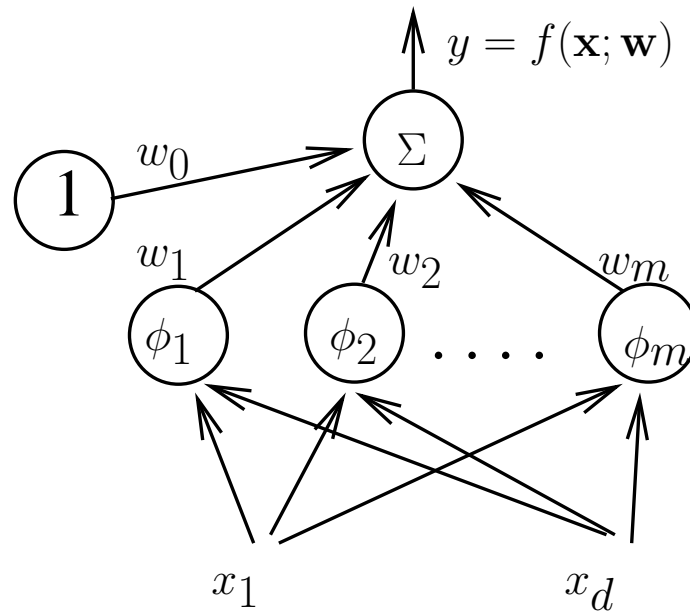
$$\phi_i(\mathbf{x}) = \begin{cases} 1 & \text{if the keyword } W_i \text{ appears in } \mathbf{x} \text{ at least once,} \\ 0 & \text{otherwise.} \end{cases}$$

- Suppose we have a set of course descriptions $\mathbf{x}_1, \dots, \mathbf{x}_N$ with y_1, \dots, y_N being course ratings by students.
- We can predict the rating using the keyword-based regression model

$$y = w_0 + \sum_{i=1}^m w_i \phi_i(\mathbf{x}).$$

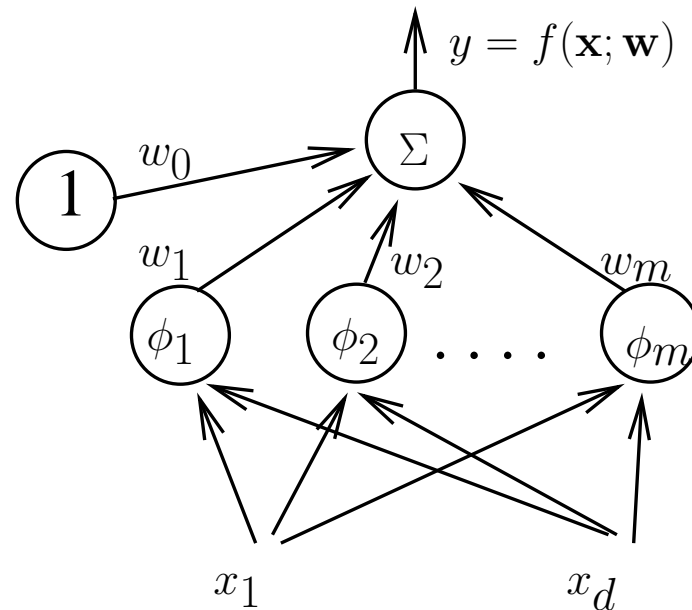
Neural networks

- A graphical representation of the general additive regression model:



- This formulation allows to learn the parameters of each basis function as well as the weights.

Learning in neural networks



- Typically, no closed-form solution as for least-squares.
- Standard method to train a NN: iterative *gradient descent*
 - Start with some (random) parameter settings.
 - *feedforward*: Predict a label for a training example
 - *feedback*: update weights so as to reduce the error on that example.

Capacity of neural networks

- What is the *capacity* of generalized linear regression? I.e., what is the class of functions that can be approximated?
- The *universal approximation theorem*:
Let basis function $\phi(\cdot)$ be a nonconstant, bounded, monotonically increasing and continuous. Suppose we have the target function f which is continuous over interval I . Then, for any $\epsilon > 0$, there exist finite M and weights w_1, \dots, w_M and $\mathbf{v}_i, i = 1, \dots$, such that

$$\max_{\mathbf{x} \in I} \left| f(\mathbf{x}) - \sum_{i=1}^M w_i \phi(\mathbf{v}_i^T \mathbf{x}) \right| < \epsilon.$$

- Note: this formulation seems to refer to a single basis function. In fact, each “internal” weight vector \mathbf{v}_i effectively defines a separate basis function of \mathbf{x} .

Regression: summary

- Standard linear regression: white Gaussian additive noise model.
- Learning through minimizing empirical squared loss
 - Equivalent to ML estimation, minimizing empirical log-loss under Gaussian model.
- Structural and approximation errors in training parametric regression on a finite data set.
- Extensions to nonlinear regression through the use of basis function.
 - Still linear in parameters \mathbf{w} .
- Dealing with overfitting: cross-validation.

Classification: a kind of regression?

- Classification problems: the labels space \mathcal{Y} is finite (discrete). E.g.,
 - Object category: horse/dog/cow/pear/tomato/apple
 - Human ID: Greg/Dan/Payman/other
 - Written digit: 0, . . . , 9
- The simplest approach: we already know how to do regression.
 - Simply predict y from \mathbf{x} using, e.g., linear least squares.

Next time

We will talk about applying what we have learned from studying regression to classification problems.