Lecture 2: Overfitting and Regularization

- Overfitting
- Cross-validation
- L2 and L1 regularization for linear estimators
- Bias-variance trade-off

Recall: Steps to solving a supervised learning problem

1. Decide what the input-output pairs are.
2. Decide how to encode inputs and outputs.

   This defines the input space $\mathcal{X}$, and the output space $\mathcal{Y}$.
3. Choose a class of hypotheses/representations $\mathcal{H}$.
4. Choose an error function (cost function) to define the best hypothesis
5. Choose an algorithm for searching efficiently through the space of hypotheses.

Recall: Linear hypothesis

- Suppose y was a linear function of $\mathbf{x}$:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 (+ \cdots)$$

- w_i are called *parameters* or *weights*
- To simplify notation, we can add an attribute $x_0 = 1$ to the other n attributes (also called *bias term* or *intercept term*):

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{n} w_i x_i = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w}$ and $\mathbf{x}$ are vectors of size $n+1$.

Recall: Least mean squares (LMS)

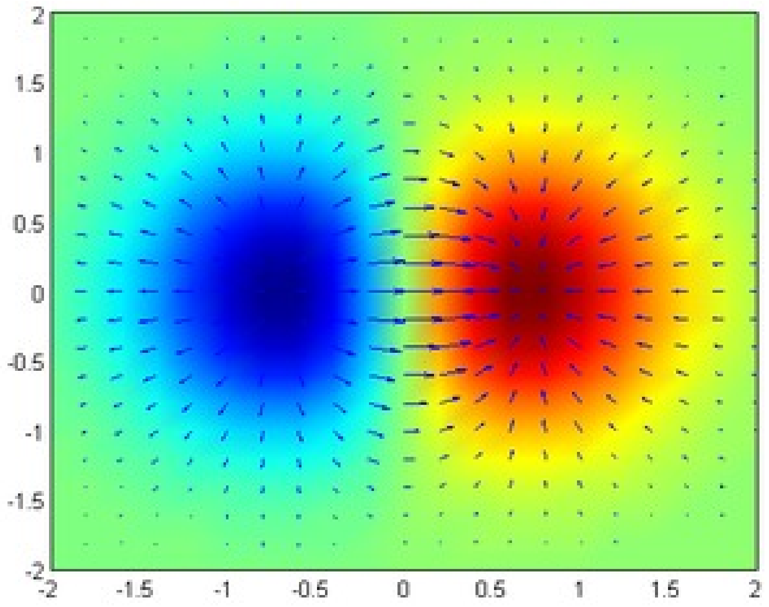
- Main idea: try to make $h_{\mathbf{w}}(\mathbf{x})$ close to y on the examples in the training set
- We define a *sum-of-squares* error function

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$$

(the $1/2$ is just for convenience)

- We want to choose $\mathbf{w}$ such as to minimize $J(\mathbf{w})$

Notation reminder: Gradient



- Multivariate generalization of the derivative.
- Points in the direction of the greatest increase of the function.

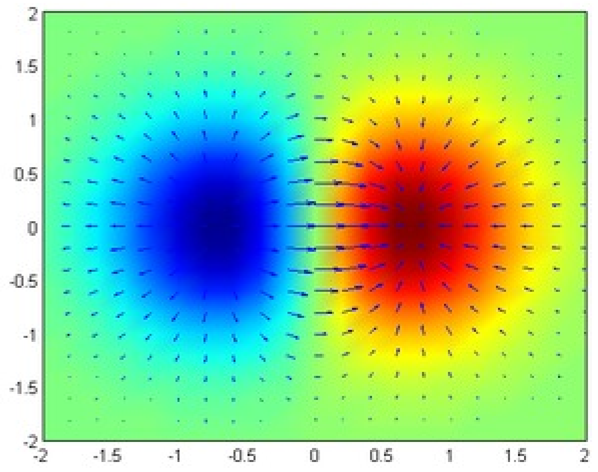
- Consider a function $f(u_1, u_2, \ldots, u_n) : \mathbb{R}^n \mapsto \mathbb{R}$ (e.g. an error function)
- The *partial derivative* w.r.t. u_i is denoted:

$$\frac{\partial}{\partial u_i} f(u_1, u_2, \ldots, u_n) : \mathbb{R}^n \mapsto \mathbb{R}$$

  The partial derivative is the derivative along the u_i axis, keeping all other variables fixed.
- The *gradient* $\nabla f(u_1, u_2, \ldots, u_n) : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a function which outputs a vector containing the partial derivatives.
  That is:

$$\nabla f = \left\langle \frac{\partial}{\partial u_1} f, \frac{\partial}{\partial u_2} f, \ldots, \frac{\partial}{\partial u_n} f \right\rangle$$

Properties of the gradient



- The inner product $\langle \nabla f(\mathbf{x}), \mathbf{v} \rangle$ between the gradient of f at $\mathbf{x}$ and any unit vector $\mathbf{v} \in \mathbb{R}^n$ is the *directional derivative* of f in the direction of $\mathbf{v}$ (i.e. the rate at which f changes at $\mathbf{x}$ in the direction $\mathbf{v}$).

$\rightarrow$ $\nabla f(\mathbf{x})$ points towards the direction of greatest increase of f at $\mathbf{x}$.

$\rightarrow$ Points such that $\nabla f(\mathbf{x}) = \mathbf{0}$ are called stationary points.

$\rightarrow$ The gradient $\nabla f(\mathbf{x})$ is orthogonal to the contour line passing through $\mathbf{x}$.

A bit of algebra

$$\begin{aligned}
\frac{\partial}{\partial w_j} J(\mathbf{w}) &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 \\
&= \frac{1}{2} \cdot 2 \sum_{i=1}^{m} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \frac{\partial}{\partial w_j} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \\
&= \sum_{i=1}^{m} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \frac{\partial}{\partial w_j} \left(\sum_{l=0}^{n} w_l x_{i,l} - y_i \right) \\
&= \sum_{i=1}^{m} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i) x_{i,j}
\end{aligned}$$

Setting all these partial derivatives to 0, we get a linear system with $(n+1)$ equations and $(n+1)$ unknowns.

The solution

- Concise form of the error function: $J(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$.
- Recalling some multivariate calculus:

$$\begin{aligned}
\nabla_{\mathbf{w}} J &= \nabla_{\mathbf{w}} \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= \frac{1}{2}\nabla_{\mathbf{w}}(\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}) \\
&= \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} = \left[\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)x_{i,j}\right]_{j=1\ldots n+1}
\end{aligned}$$

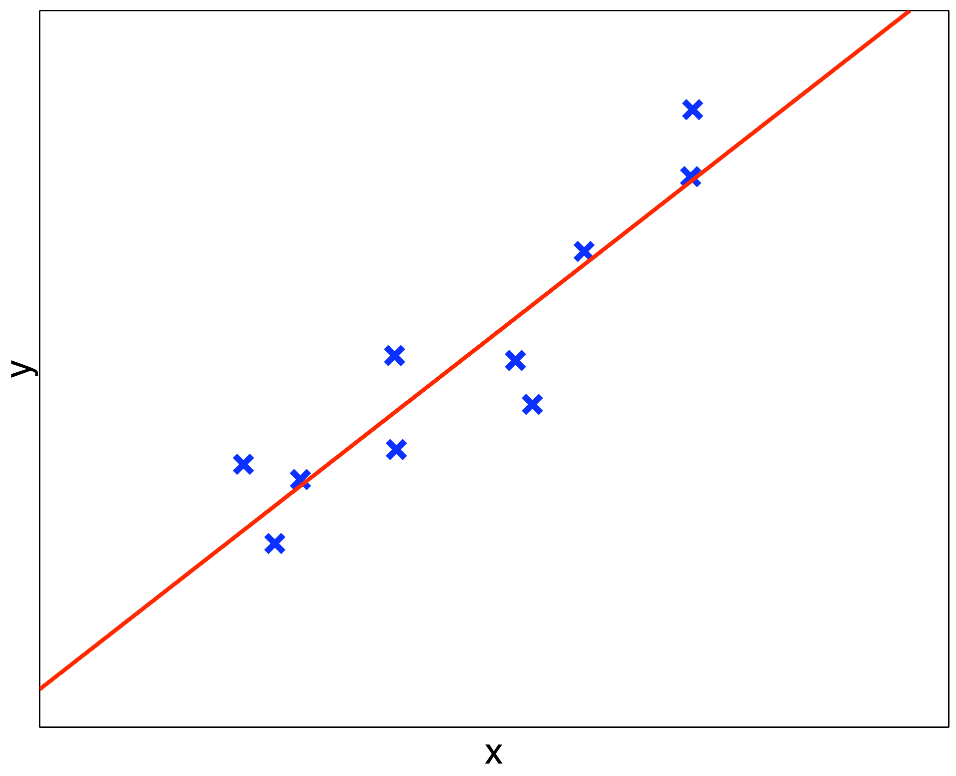
- Setting gradient equal to zero:

$$\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} = 0 \qquad \Rightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y} \qquad \Rightarrow \mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- The inverse exists if the columns of $\mathbf{X}$ are linearly independent.

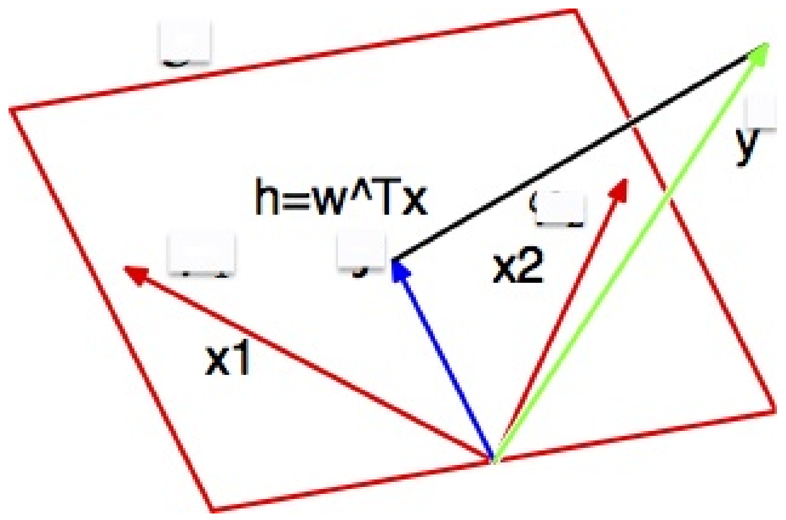
Example: Data and best linear hypothesis

$$y = 1.60x + 1.05$$



Coming back to mean-squared error function...

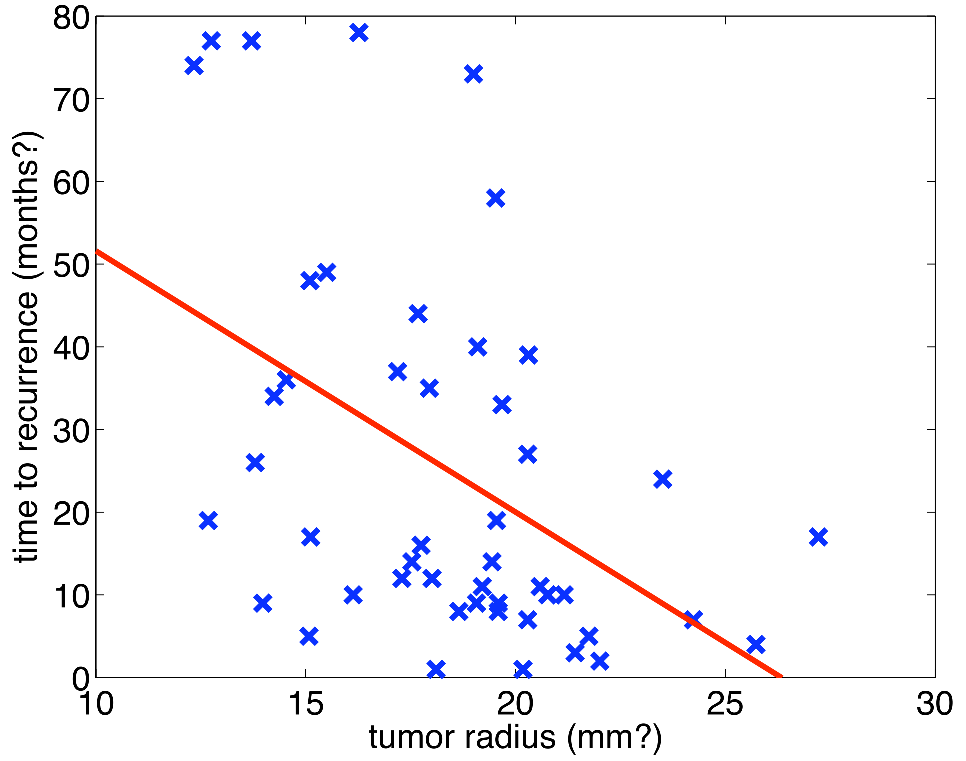
- Good intuitive feel (small errors are ignored, large errors are penalized)
- Nice math (closed-form solution, unique global optimum)
- Geometric interpretation



Linear regression summary

- The optimal solution (minimizing sum-squared-error) can be computed in polynomial time in the size of the data set.
- The solution is $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, where $\mathbf{X}$ is the data matrix augmented with a column of ones, and $\mathbf{y}$ is the column vector of target outputs.
- A very rare case in which an analytical, exact solution is possible

Predicting recurrence time based on tumor size



Is linear regression enough?

- Linear regression is too simple for most realistic problems

  But it should be the first thing you try for real-valued outputs!

- Two possible solutions:

  1. Transform the data
     - Add cross-terms, higher-order terms
     - More generally, apply a transformation of the inputs from $\mathcal{X}$ to some other space $\mathcal{X}'$, then do linear regression in the transformed space
  2. Use a different hypothesis class (e.g. non-linear functions)

- Today we focus on the first approach

Polynomial fits

- Suppose we want to fit a higher-degree polynomial to the data (e.g., $y = w_0 + w_1 x^1 + w_2 x^2$).
- Suppose for now that there is a single input variable per training sample.
- How do we do it?

Answer: Polynomial regression

- Given data: $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$.
- Suppose we want a degree-d polynomial fit for the data

$$\mathbf{X} = [x_1, \cdots, x_m]^\top, \quad \mathbf{y} = [y_1, \cdots, y_m]^\top$$

- Let $\mathbf{y}$ be as before and let

$$\mathbf{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & & \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^d \end{bmatrix}$$

- Solve the linear regression $\mathbf{\Phi w} \approx \mathbf{y}$.

Linear function approximation in general

- The best $\mathbf{w}$ is considered the one which minimizes the sum-squared error over the training data:

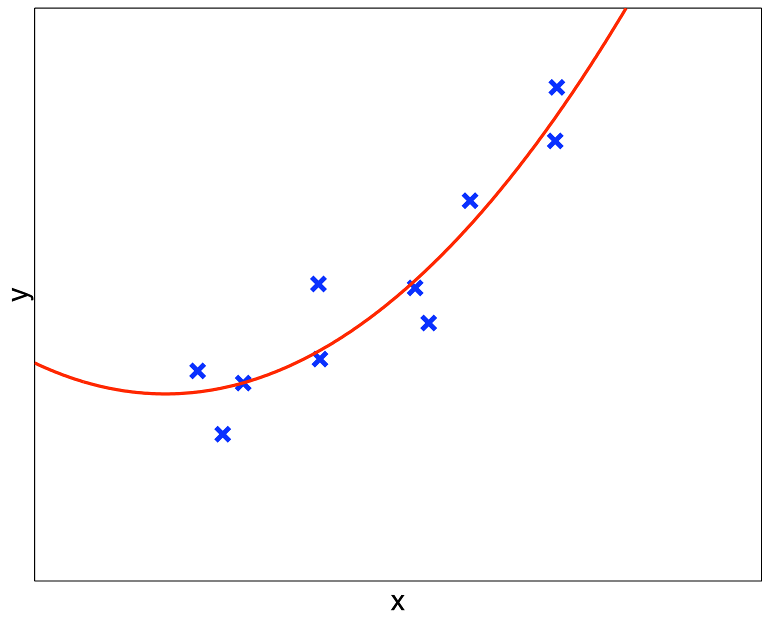
$$\sum_{i=1}^{m}(y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2$$

- We can find the best $\mathbf{w}$ in closed form:

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

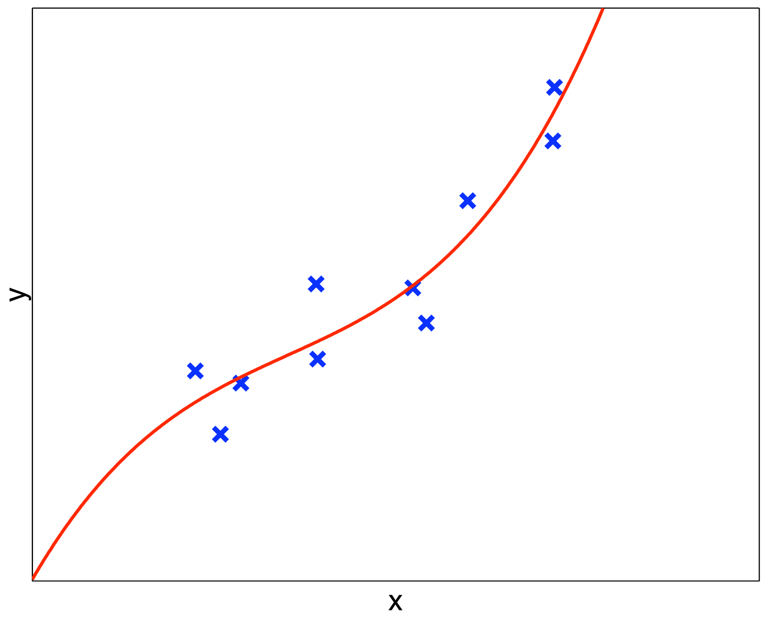
- By linear models, we mean that the hypothesis function $h_{\mathbf{w}}(\mathbf{x})$ is a linear function of the parameters $\mathbf{w}$

Order-2 fit



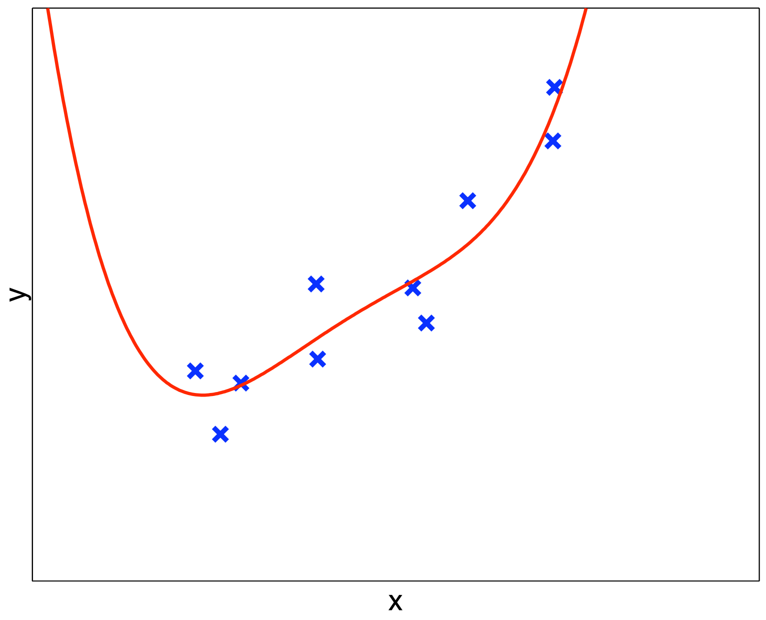
Is this a better fit to the data?

Order-3 fit



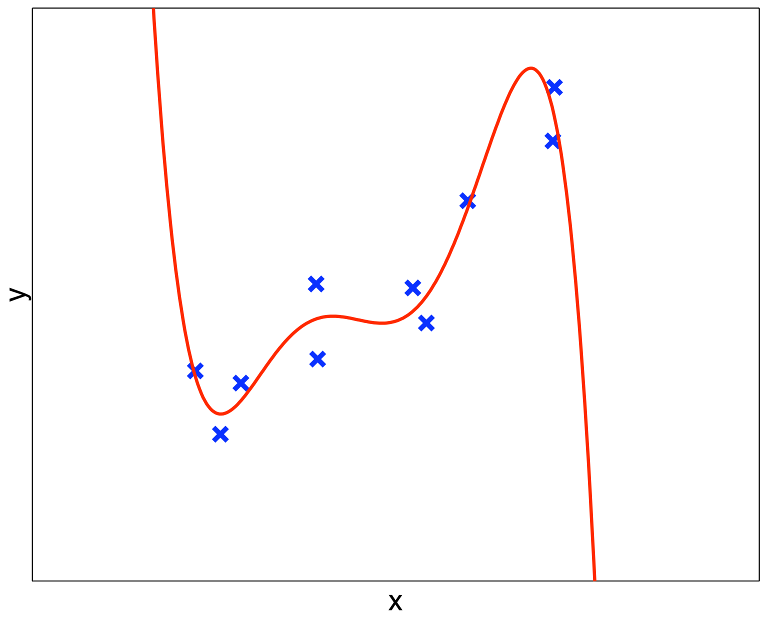
Is this a better fit to the data?

Order-4 fit



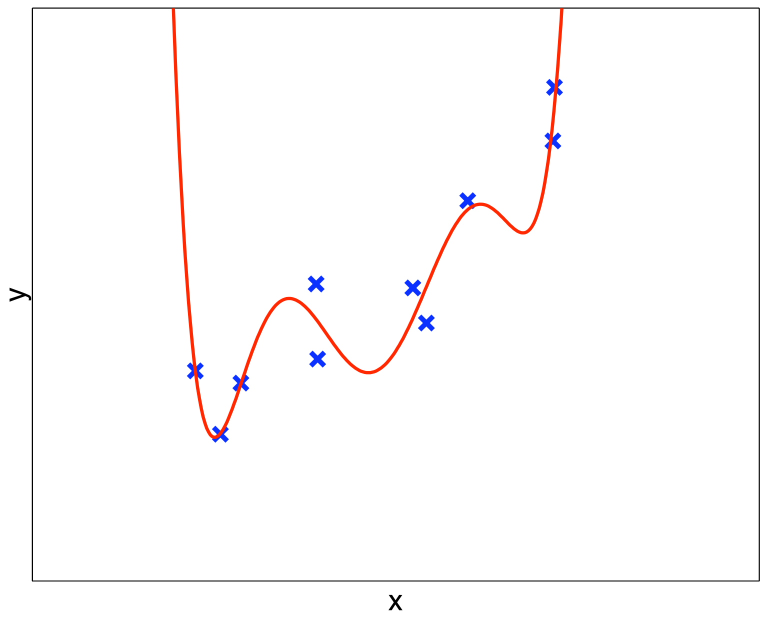
Is this a better fit to the data?

Order-5 fit



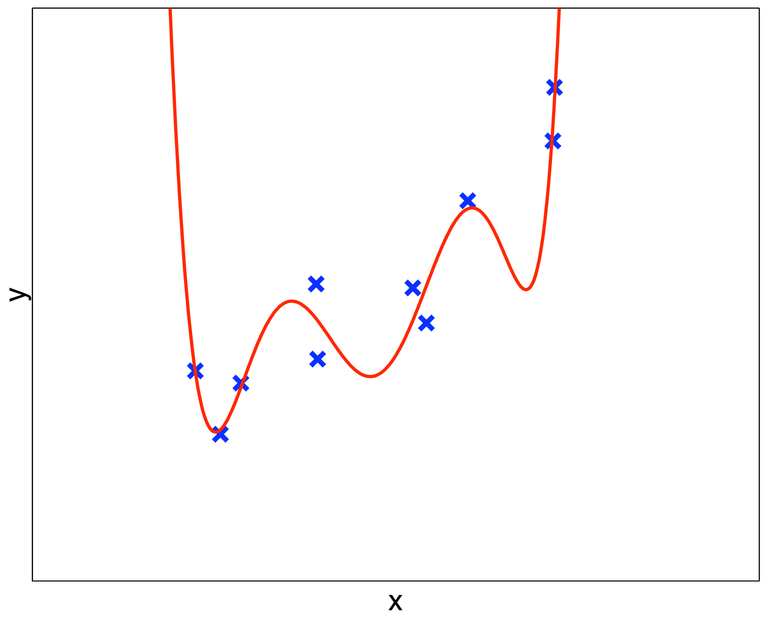
Is this a better fit to the data?

Order-6 fit



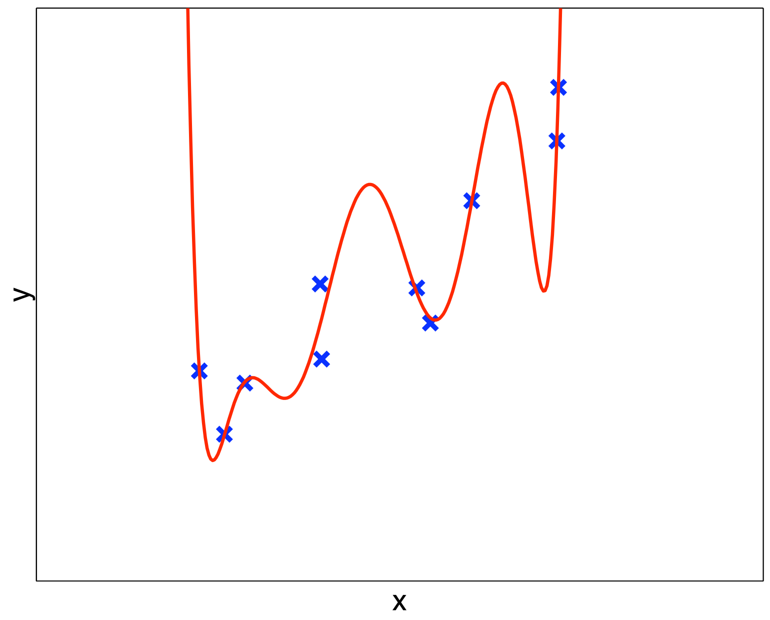
Is this a better fit to the data?

Order-7 fit



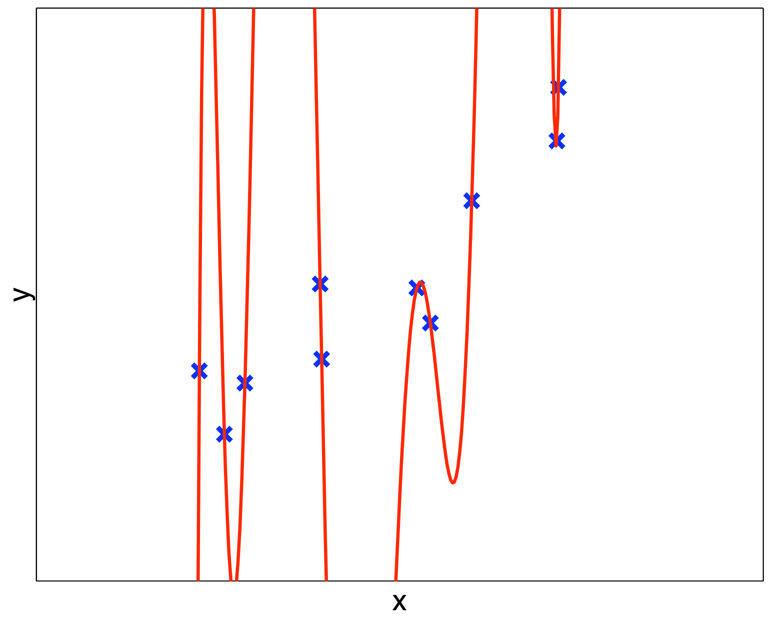
Is this a better fit to the data?

Order-8 fit



Is this a better fit to the data?

Order-9 fit

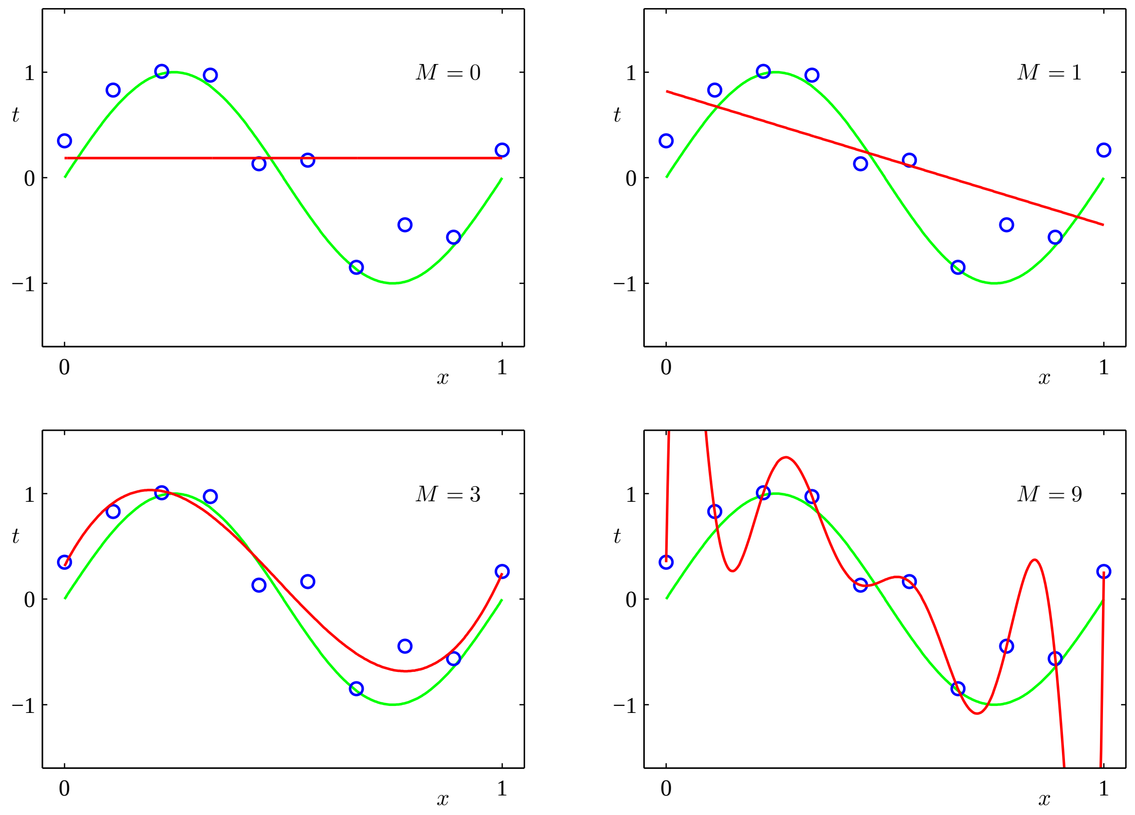


Is this a better fit to the data?

Overfitting

- A general, *HUGELY IMPORTANT* problem for all machine learning algorithms
- We can find a hypothesis that predicts perfectly the training data but *does not generalize* well to new data
- E.g., a lookup table!
- We are seeing an instance here: if we have a lot of parameters, the hypothesis *memorizes* the data points, but is wild everywhere else.

Another overfitting example

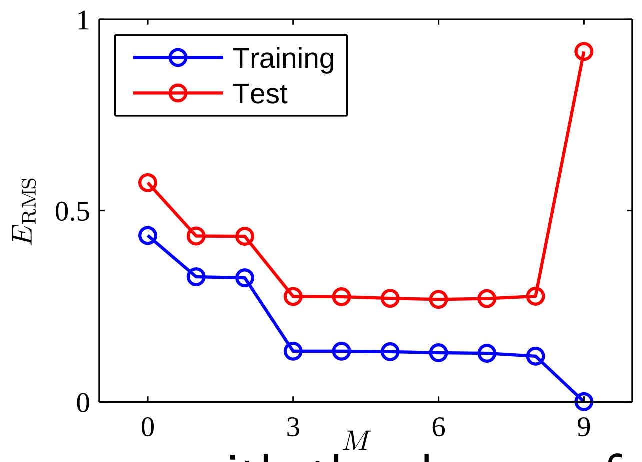


- The higher the degree of the polynomial M, the more degrees of freedom, and the more capacity to “overfit” the training data
- Typical overfitting means that error on the training data is very low, but error on new instances is high

Overfitting more formally

- Assume that the data is drawn from some fixed, unknown probability distribution
- Every hypothesis has a *true* error $J^*(h)$, which is the expected error when data is drawn from the distribution.
- Because we do not have all the data, we measure the error on the training set $J_D(h)$
- Suppose we compare hypotheses h_1 and h_2 on the training set, and $J_D(h_1) < J_D(h_2)$
- If h_2 is *truly* better, i.e. $J^*(h_2) < J^*(h_1)$, our algorithm is overfitting.
- We need theoretical and empirical methods to guard against it!

Typical overfitting plot



- The training error decreases with the degree of the polynomial M, i.e. *the complexity of the hypothesis*
- The testing error, measured on independent data, decreases at first, then starts increasing
- Cross-validation helps us:
  - Find a good hypothesis class (M in our case), using a *validation set of data*
  - Report unbiased results, using a *test set*, untouched during either parameter training or validation

Cross-validation

- A general procedure for estimating the true error of a predictor
- The available data is split into two subsets:
  - A *training and validation set* used only to find the right predictor
  - A *test set* used to report the prediction error of the algorithm
- These sets *must be disjoint*!
- The process is repeated several times, and the results are averaged to provide error estimates.

Model selection with leave-one-out cross-validation

1. For each order of polynomial, d:
   (a) Repeat the following procedure m times:
      i. Leave out *ith instance* from the training set, to estimate the true prediction error; we will put it in a *validation set*
      ii. Use all the other instances to find best parameter vector, $\mathbf{w}_{d,i}$
      iii. Measure the error in predicting the label on the instance left out, for the $\mathbf{w}_{d,i}$ parameter vector; call this $J_{d,i}$
      iv. This is a *(mostly) unbiased estimate of the true prediction error*

   (b) Compute the average of the estimated errors: $J_d = \frac{1}{m}\sum_{i=1}^{m} J_{d,i}$
2. Choose the d with lowest average estimated error: $d^* = \arg\min_d J(d)$

Summary of leave-one-out cross-validation

- A very easy to implement algorithm
- Provides a great estimate of the true error of a predictor
- Computational cost scales with the number of instances (examples), so it can be prohibitive, especially if finding the best predictor is expensive
- Alternatives:
  - Leave-k-out generalizes LOO, computationally very expensive.
  - k-fold cross-validation: split the data set into k parts, then proceed as above.

Regularization

- Remember the intuition: complicated hypotheses lead to overfitting
- Idea: change the error function to *penalize hypothesis complexity*:

$$J(\mathbf{w}) = J_D(\mathbf{w}) + \lambda J_{pen}(\mathbf{w})$$

  This is called *regularization* in machine learning and *shrinkage* in statistics
- λ is called *regularization coefficient* and controls how much we value fitting the data well, vs. a simple hypothesis

L_2 regularization for linear models

- *L_2 regularization* (or *weight decay* in neural networks): add a squared penalty on the weights:

$$J_\lambda(\mathbf{w}) = \frac{1}{2}(\mathbf{\Phi w} - \mathbf{y})^\top(\mathbf{\Phi w} - \mathbf{y}) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

- Why?
  - A simple hypothesis should not be too sensitive to small perturbation of the input
  - Math works out nicely
  - Resolve the issue of $\mathbf{\Phi}^\top\mathbf{\Phi}$ not being invertible in linear regression, e.g. large n small m (original motivation)...

L_2 regularization: closed from solution

$$J_\lambda(\mathbf{w}) = \frac{1}{2}(\mathbf{\Phi w} - \mathbf{y})^\top(\mathbf{\Phi w} - \mathbf{y}) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

- By re-grouping terms, we get:

$$J_\lambda(\mathbf{w}) = \frac{1}{2}\left(\mathbf{w}^\top(\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})\mathbf{w} - \mathbf{w}^\top\mathbf{\Phi}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{\Phi w} + \mathbf{y}^\top\mathbf{y}\right)$$

- Optimal solution (obtained by solving $\nabla J_\lambda(\mathbf{w}) = 0$)

$$\mathbf{w} = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{y}$$

(observe that $\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I}$ is invertible)

Side note: data centering

- For linear regression, we incorporated the bias term in $\mathbf{w} \in \mathbb{R}^{n+1}$ by adding a 1 to each input vector $\mathbf{x} \in \mathbb{R}^n$. Why cannot we do the same for regression with L_2 regularization?
- Instead we center the data, i.e. remove means from inputs and outputs:
  - Let $\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i$ and $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$.
  - Consider the new data set $\{(\mathbf{x}_i - \bar{\mathbf{x}}, y_i - \bar{y})\}_{i=1}^{m}$.
  - Solving the regularized regression problem on this new dataset is *equivalent* to solving the original problem
  - You can check that the bias term of the solution on this new data set is 0...

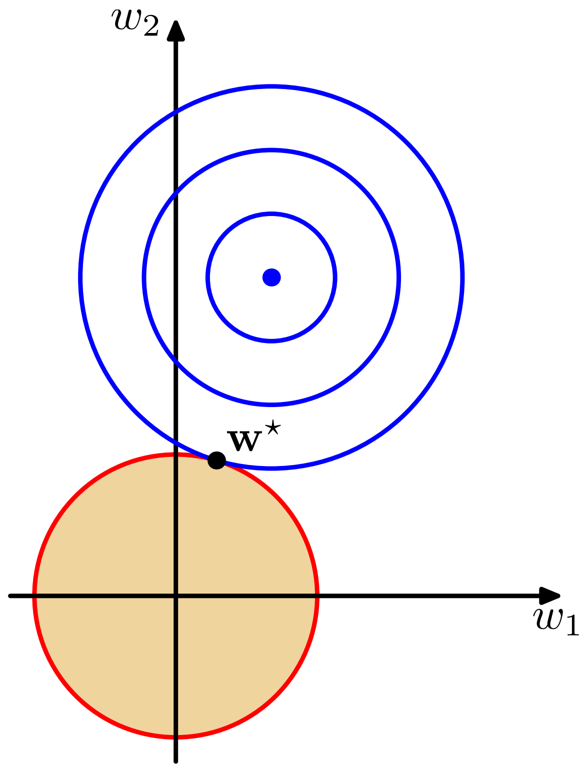
  $\hookrightarrow$ exercise
- Also make sure that columns of $\mathbf{X}$ are on the same scale (e.g. normalize)

What L_2 regularization does

$$\arg\min_{\mathbf{w}} \frac{1}{2}(\mathbf{\Phi w} - \mathbf{y})^\top(\mathbf{\Phi w} - \mathbf{y}) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{y}$$

- If $\lambda = 0$, the solution is the same as in regular least-squares linear regression
- If $\lambda \rightarrow \infty$, the solution $\mathbf{w} \rightarrow 0$
- Positive λ will cause the magnitude of the weights to be smaller than in the usual linear solution
- This is also known as *ridge regression*, and a special case of Tikhonov regularization

Visualizing regularization (2 parameters)



$$\mathbf{w}^* = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}\mathbf{y}$$

Pros and cons of L_2 regularization

- If λ is at a "good" value, regularization helps to avoid overfitting
- Choosing λ may be hard: cross-validation is often used
- If there are irrelevant features in the input (i.e. features that do not affect the output), L_2 will give them small, but non-zero weights.
- Ideally, irrelevant input should have weights exactly equal to 0.

L_1 Regularization for linear models

- Instead of requiring the L_2 norm of the weight vector to be bounded, make the requirement on the L_1 norm:

$$\begin{aligned} \min_{\mathbf{w}} J_D(\mathbf{w}) &= \min_{\mathbf{w}} (\mathbf{\Phi}\mathbf{w} - \mathbf{y})^\top (\mathbf{\Phi}\mathbf{w} - \mathbf{y}) \\ \text{such that } \sum_{i=1}^{n} |w_i| &\leq \eta \end{aligned}$$

- This yields an algorithm called Lasso (Tibshirani, 1996)

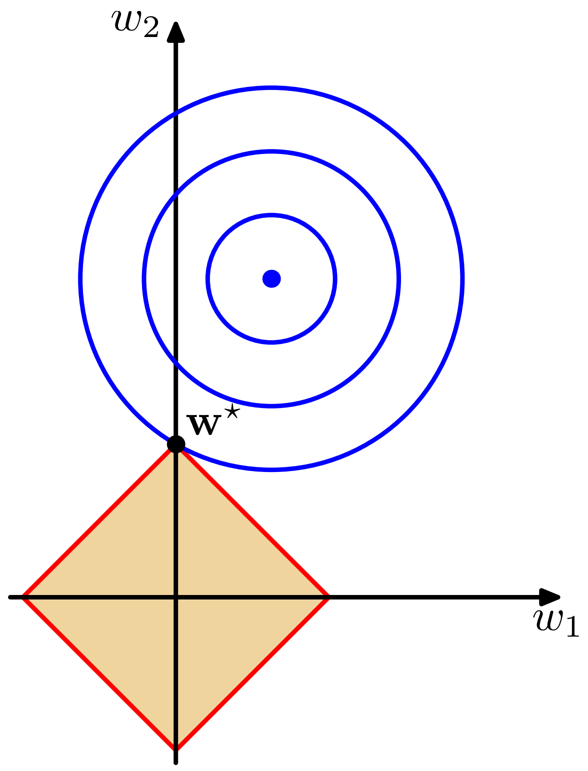
Solving L_1 regularization

- The optimization problem is a quadratic program
- There is one constraint for each possible sign of the weights (2^n constraints for n weights)
- For example, with two weights:

$$\begin{aligned} \min_{w_1,w_2} \quad & \sum_{j=1}^{m}(y_j - w_1x_1 - w_2x_2)^2 \\ \text{such that } w_1 + w_2 &\leq \eta \\ w_1 - w_2 &\leq \eta \\ -w_1 + w_2 &\leq \eta \\ -w_1 - w_2 &\leq \eta \end{aligned}$$

- Solving this program directly can be done for problems with a small number of inputs

Visualizing L_1 regularization

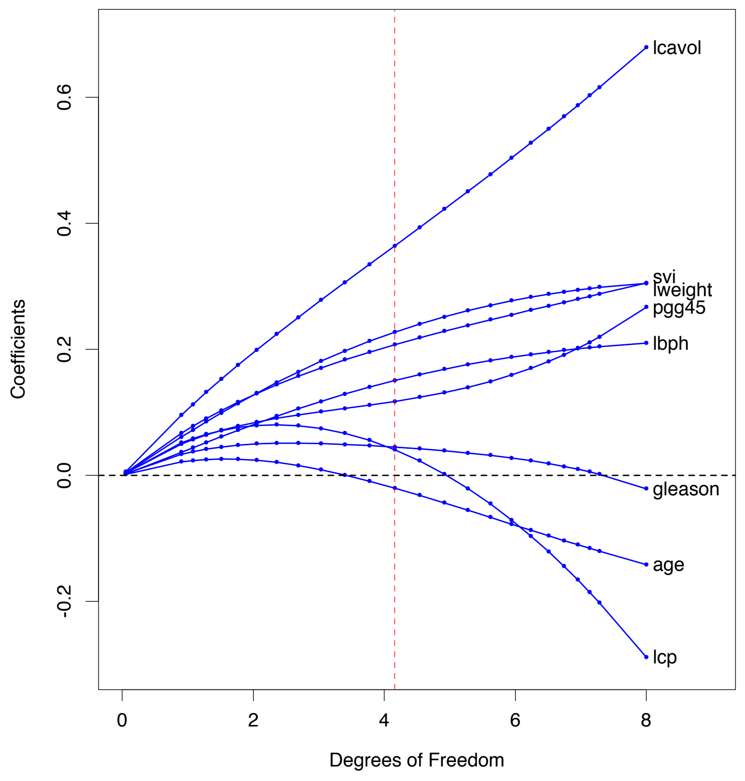


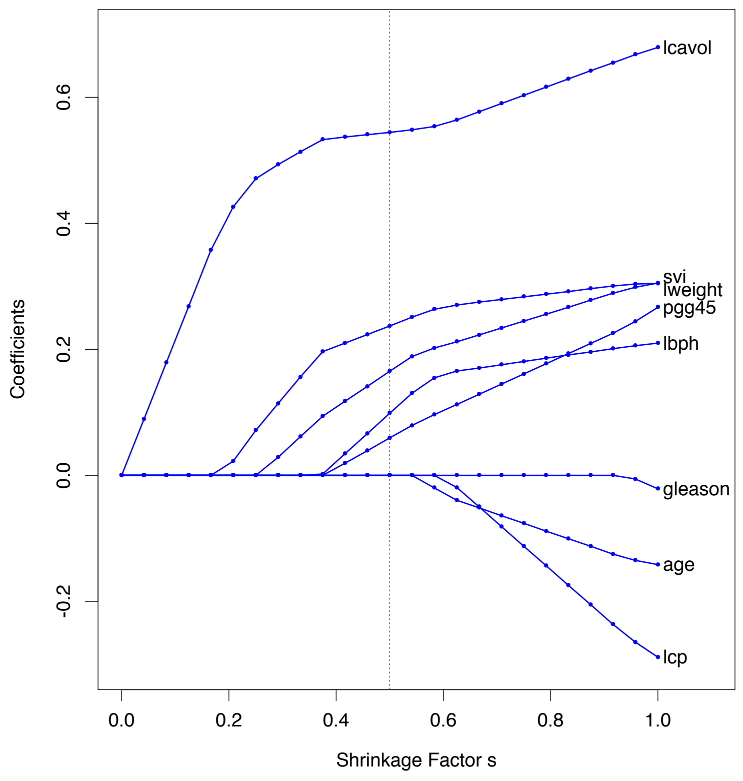
- If λ is big enough, the circle is very likely to intersect the diamond at one of the corners
- This makes L_1 regularization much more likely to make some weights *exactly* 0

Pros and cons of L_1 regularization

- If there are irrelevant input features, Lasso is likely to make their weights 0, while L_2 is likely to just make all weights small
- Lasso is biased towards providing *sparse solutions* in general
- Lasso optimization is computationally more expensive than L_2
- More efficient solution methods have to be used for large numbers of inputs (e.g. least-angle regression, 2003).
- L_1 methods of various types are very popular
- One can combine L_1 and L_2 regularization (elastic-net)

Example of L1 vs L2 effect





- Note the sparsity in the coefficients induces by L_1

The anatomy of the error of an estimator

- Suppose we have a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $y = f(\mathbf{x}) + \epsilon$ and ϵ is Gaussian noise with zero mean and standard deviation σ^2
- We fit a linear hypothesis $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, such as to minimize sum-squared error over the training data:

$$\sum_{i=1}^{m} (y_i - h(\mathbf{x}_i))^2$$

- Because of the hypothesis class that we chose (hypotheses linear in the parameters) for some target functions f we will have a *systematic prediction error*
- Even if f were truly from the hypothesis class we picked, depending on the data set we have, the parameters $\mathbf{w}$ that we find may be different; this *variability* due to the specific data set on hand is a different source of error

Bias-variance analysis

- Given a new data point $\mathbf{x}$, what is the *expected prediction error*?
- Assume that the data points are drawn *independently and identically distributed (i.i.d.)* from a unique underlying probability distribution $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$
- The goal of the analysis is to compute, for an arbitrary given point $\mathbf{x}$,

$$E_P\left[(y - h(\mathbf{x}))^2|\mathbf{x}\right]$$

  where the expectation is over all training sets of a given size drawn according to $P(\cdot, \cdot)$ and over y drawn according to $P(\cdot|\mathbf{x})$.
- For a given hypothesis class, we can also compute the *true error*, which is the expected error over the input distribution:

$$\sum_{\mathbf{x}} E_P\left[(y - h(\mathbf{x}))^2|\mathbf{x}\right] P(\mathbf{x})$$

  (if $\mathbf{x}$ continuous, sum becomes integral with appropriate conditions).
- We will decompose this expectation into three components

Recall: Statistics 101

- Let X be a random variable with possible values $x_i, i = 1 \dots n$ and with probability distribution $P(X)$
- The *expected value* or *mean* of X is:

$$E[X] = \sum_{i=1}^{n} x_i P(x_i)$$

- If X is continuous, roughly speaking, the sum is replaced by an integral, and the distribution by a density function
- The *variance* of X is:

$$\begin{aligned} Var[X] &= E[(X - E(X))^2] \\ &= E[X^2] - (E[X])^2 \end{aligned}$$

The variance lemma

$$\begin{aligned}
Var[X] &= E[(X - E[X])^2] \\
&= \sum_{i=1}^{n} (x_i - E[X])^2 P(x_i) \\
&= \sum_{i=1}^{n} (x_i^2 - 2x_i E[X] + (E[X])^2) P(x_i) \\
&= \sum_{i=1}^{n} x_i^2 P(x_i) - 2E[X] \sum_{i=1}^{n} x_i P(x_i) + (E[X])^2 \sum_{i=1}^{n} P(x_i) \\
&= E[X^2] - 2E[X]E[X] + (E[X])^2 \cdot 1 \\
&= E[X^2] - (E[X])^2
\end{aligned}$$

We will use the form:

$$E[X^2] = (E[X])^2 + Var[X]$$

Bias-variance decomposition

- Simple algebra:

$$\begin{aligned} E_P\left[(y-h(\mathbf{x}))^2|\mathbf{x}\right] &= E_P\left[(h(\mathbf{x}))^2 - 2yh(\mathbf{x}) + y^2|\mathbf{x}\right] \\ &= E_P\left[(h(\mathbf{x}))^2|\mathbf{x}\right] + E_P\left[y^2|\mathbf{x}\right] - 2E_P[y|\mathbf{x}]E_P\left[h(\mathbf{x})|\mathbf{x}\right] \end{aligned}$$

- Let $\bar{h}(\mathbf{x}) = E_P[h(\mathbf{x})|\mathbf{x}]$ denote the *mean prediction* of the hypothesis at $\mathbf{x}$, when h is trained with data drawn from P
- For the first term, using the variance lemma, we have:

$$E_P[(h(\mathbf{x}))^2|\mathbf{x}] = E_P[(h(\mathbf{x}) - \bar{h}(\mathbf{x}))^2|\mathbf{x}] + (\bar{h}(\mathbf{x}))^2$$

- Note that $E_P[y|\mathbf{x}] = E_P[f(\mathbf{x}) + \epsilon|\mathbf{x}] = f(\mathbf{x})$ (because of linearity of expectation and the assumption on $\epsilon \sim \mathcal{N}(0, \sigma)$)
- For the second term, using the variance lemma, we have:

$$E[y^2|\mathbf{x}] = E[(y - f(\mathbf{x}))^2|\mathbf{x}] + (f(\mathbf{x}))^2$$

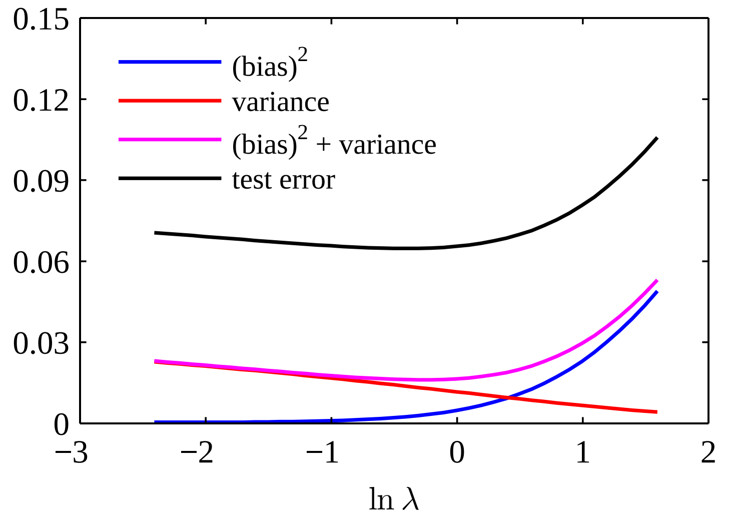
Bias-variance decomposition (2)

- Putting everything together, we have:

$$\begin{aligned} E_P\left[(y - h(\mathbf{x}))^2|\mathbf{x}\right] &= E_P[(h(\mathbf{x}) - \bar{h}(\mathbf{x}))^2|\mathbf{x}] + (\bar{h}(\mathbf{x}))^2 - 2f(\mathbf{x})\bar{h}(\mathbf{x}) \\ &\quad + E_P[(y - f(\mathbf{x}))^2|\mathbf{x}] + (f(\mathbf{x}))^2 \\ &= E_P[(h(\mathbf{x}) - \bar{h}(\mathbf{x}))^2|\mathbf{x}] \quad + \quad (f(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \quad + \quad E[(y - f(\mathbf{x}))^2|\mathbf{x}] \end{aligned}$$

- The first term, $E_P[(h(\mathbf{x}) - \bar{h}(\mathbf{x}))^2|\mathbf{x}]$, is the *variance* of the hypothesis h at $\mathbf{x}$, when trained with finite data sets sampled randomly from P
- The second term, $(f(\mathbf{x}) - \bar{h}(\mathbf{x}))^2$, is the *squared bias* (or systematic error) which is associated with the class of hypotheses we are considering
- The last term, $E[(y - f(\mathbf{x}))^2|\mathbf{x}]$ is the *noise*, which is due to the problem at hand, and cannot be avoided

Error decomposition



- The bias-variance sum approximates well the test error over a set of 1000 points
- x-axis measures the hypothesis complexity (decreasing left-to-right)
- Simple hypotheses usually have high bias (bias will be high at many points, so it will likely be high for many possible input distributions)
- Complex hypotheses have high variance: the hypothesis is very dependent on the data set on which it was trained.

Bias-variance trade-off

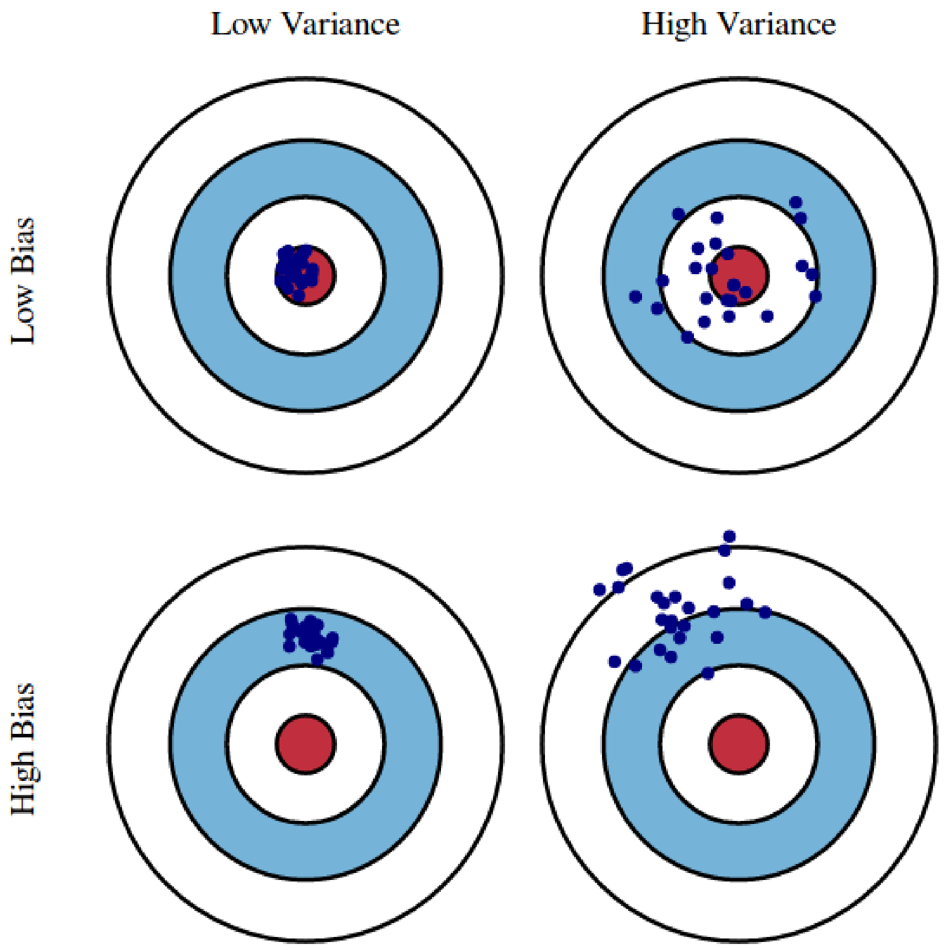


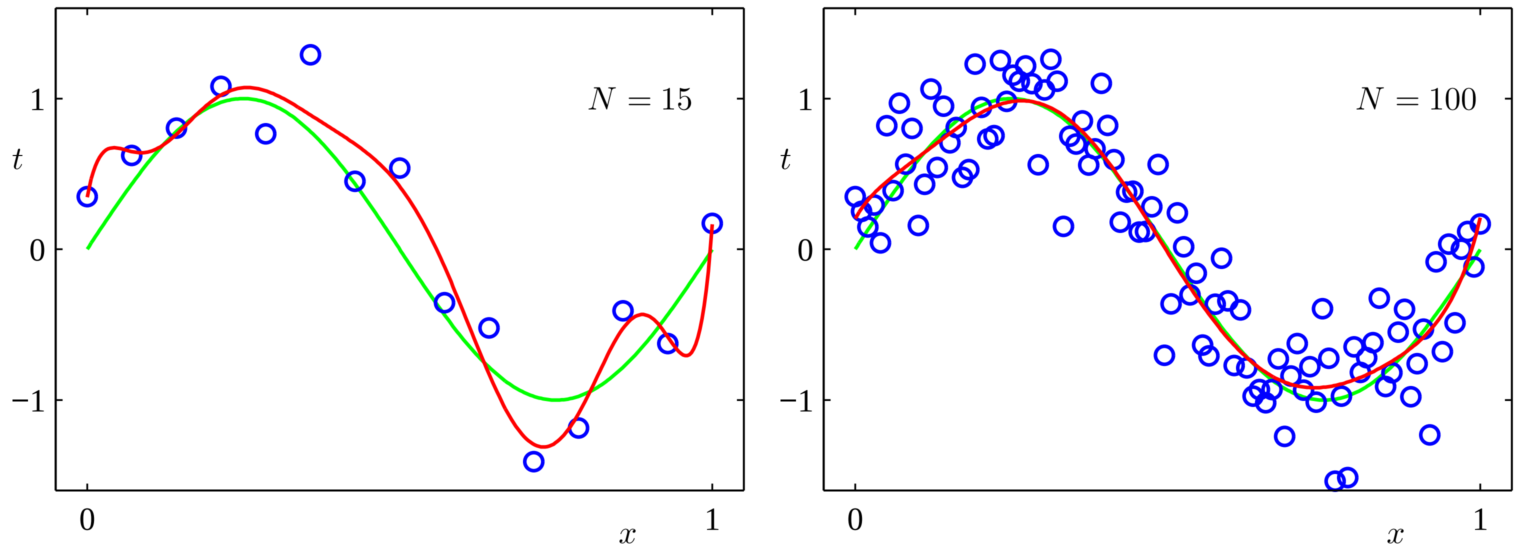
image credit: Scott Fortman-roe (http://scott.fortmann-roe.com/docs/BiasVariance.html)

Bias-variance trade-off

- Typically, bias comes from not having good hypotheses in the considered class
- Variance results from the hypothesis class containing “too many” hypotheses
- Hence, we are faced with a *trade-off*: choose a more expressive class of hypotheses, which will generate higher variance, or a less expressive class, which will generate higher bias
- Making the trade-off has to depend on the amount of data available to fit the parameters (data usually mitigates the variance problem)

More on overfitting

- Overfitting depends on the amount of data, relative to the complexity of the hypothesis
- With more data, we can explore more complex hypotheses spaces, and still find a good solution



$N = 15$
$N = 100$
t
x
1
0
−1
0
1