Lecture 3: More on linear methods for regression

- Why least-squares? A probabilistic analysis
- L2 and L1 regularization for linear estimators
- Bayesian learning and regularization

Recall: Linear function approximation

- Given a set of examples $\langle \mathbf{x}_i, y_i \rangle_{i=1...m}$, we fit a hypothesis

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{k=0}^{K-1} w_k \phi_k(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

  where ϕ_k are called basis functions
- The best $\mathbf{w}$ is considered the one which minimizes the sum-squared error over the training data:

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

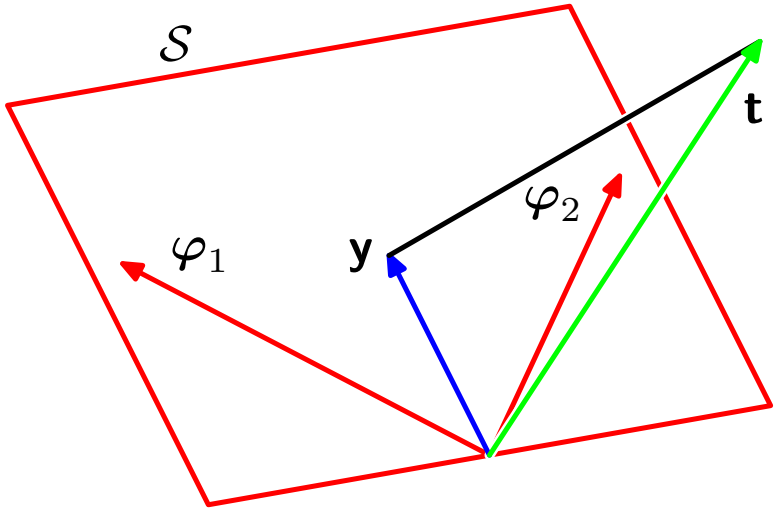
- We can find the best $\mathbf{w}$ in closed form:

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

  or by gradient descent (if we want to avoid the matrix inversion)

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 (in our notation, $\mathbf{t}$ is $\mathbf{y}$ and $\mathbf{y}$ is $h_{\mathbf{w}}(\mathbf{x})$)



- Any other interpretation?

A probabilistic assumption

- Assume y_i is a noisy target value, generated from a hypothesis $h_{\mathbf{w}}(\mathbf{x})$
- More specifically, assume that there exists $\mathbf{w}$ such that:

$$y_i = h_{\mathbf{w}}(\mathbf{x}_i) + \epsilon_i$$

  where ϵ_i is random variable (noise) drawn independently for each $\mathbf{x}_i$ according to some Gaussian (normal) distribution with mean zero and variance σ.
- How should we choose the parameter vector $\mathbf{w}$?

Bayes theorem in learning

Let h be a hypothesis and D be the set of training data.
Using Bayes theorem, we have:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)},$$

where:

- $P(h)$ is the *prior probability of hypothesis* h
- $P(D) = \int_h P(D|h)P(h)$ is the probability of training data D (normalization, independent of h)
- $P(h|D)$ is the probability of h given D
- $P(D|h)$ is the probability of D given h (*likelihood of the data*)

Choosing hypotheses

- What is the most probable hypothesis given the training data?
- *Maximum a posteriori (MAP)* hypothesis h_{MAP}:

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(h|D) \\ &= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \text{(using Bayes theorem)} \\ &= \arg\max_{h \in H} P(D|h)P(h) \end{aligned}$$

  Last step is because $P(D)$ is independent of h (so constant for the maximization)

- This is the Bayesian answer (more in a minute)

Maximum likelihood estimation

$$h_{MAP} = \arg\max_{h \in H} P(D|h)P(h)$$

- If we assume $P(h_i) = P(h_j)$ (all hypotheses are equally likely a priori) then we can further simplify, and choose the *maximum likelihood (ML) hypothesis*:

$$h_{ML} = \arg\max_{h \in H} P(D|h) = \arg\max_{h \in H} L(h)$$

- Standard assumption: the training examples are *independently identically distributed (i.i.d.)*
- This alows us to simplify $P(D|h)$:

$$P(D|h) = \prod_{i=1}^{m} P(\langle \mathbf{x_i}, y_i \rangle | h) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i; h)P(\mathbf{x}_i)$$

The $\log$ trick

- We want to maximize:

$$L(h) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i; h)P(\mathbf{x}_i)$$

  This is a product, and products are hard to maximize!
- Instead, we will maximize $\log L(h)$! (the log-likelihood function)

$$\log L(h) = \sum_{i=1}^{m} \log P(y_i|\mathbf{x}_i; h) + \sum_{i=1}^{m} \log P(\mathbf{x}_i)$$

- The second sum depends on D, but not on h, so it can be ignored in the search for a good hypothesis

Maximum likelihood for regression

- Adopt the assumption that:

$$y_i = h_{\mathbf{w}}(\mathbf{x}_i) + \epsilon_i,$$

  where $\epsilon_i \sim \mathcal{N}(0, \sigma)$.
- The best hypothesis maximizes the likelihood of $y_i - h_{\mathbf{w}}(\mathbf{x}_i) = \epsilon_i$
- Hence,

$$L(\mathbf{w}) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_i - h_{\mathbf{w}}(\mathbf{x}_i)}{\sigma}\right)^2}$$

  because the noise variables ϵ_i are from a Gaussian distribution

Applying the $\log$ trick

$$\begin{aligned}\log L(\mathbf{w}) &= \sum_{i=1}^{m} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2}{\sigma^2}} \right) \\ &= \sum_{i=1}^{m} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{m} \frac{1}{2} \frac{(y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2}{\sigma^2}\end{aligned}$$

Maximizing the right hand side is the same as minimizing:

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

This is our old friend, the sum-squared-error function! (the constants that are independent of h can again be ignored)

Maximum likelihood hypothesis for least-squares estimators

- Under the assumption that the training examples are i.i.d. and that we have *Gaussian target noise*, the maximum likelihood parameters $\mathbf{w}$ are those minimizing the sum squared error:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} (y_i - h_{\mathbf{w}}(\mathbf{x_i}))^2$$

- This makes explicit the hypothesis behind minimizing the sum-squared error
- If the noise is not normally distributed, maximizing the likelihood will not be the same as minimizing the sum-squared error (see homework 2)
- In practice, different loss functions may be needed

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

Regularization for linear models

- A squared penalty on the weights would make the math work nicely in our case:

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

- This is also known as L_2 *regularization*, or *weight decay* in neural networks
- By re-grouping terms, we get:

$$J_D(\mathbf{w}) = \frac{1}{2}(\mathbf{w}^T(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})\mathbf{w} - \mathbf{w}^T\mathbf{\Phi}^T\mathbf{y} - \mathbf{y}^T\mathbf{\Phi w} + \mathbf{y}^T\mathbf{y})$$

- Optimal solution (obtained by solving $\nabla_{\mathbf{w}} J_D(\mathbf{w}) = 0$)

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

What L_2 regularization does

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

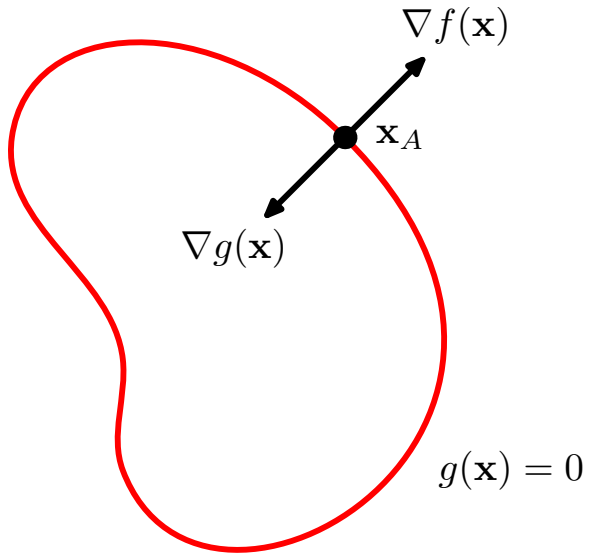
- If $\lambda = 0$, the solution is the same as in regular least-squares linear regression
- If $\lambda \to \infty$, the solution $\mathbf{w} \to 0$
- Positive λ will cause the magnitude of the weights to be smaller than in the usual linear solution
- A different view of regularization: we want to optimize the error while keeping the L_2 norm of the weights, $\mathbf{w}^T\mathbf{w}$, bounded.

Detour: Constrained optimization

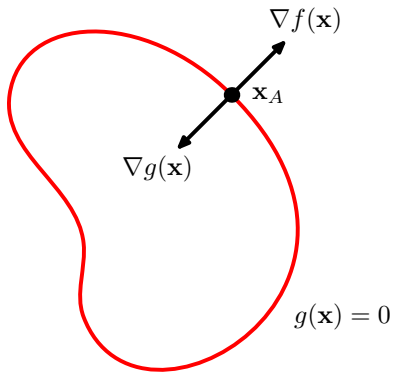
Suppose we want to find

$$\min_{\mathbf{w}} f(\mathbf{w})$$

$$\text{such that } g(\mathbf{w}) = 0$$



Detour: Lagrange multipliers

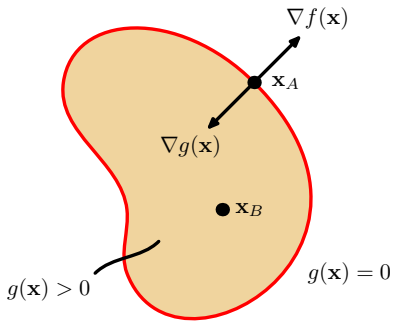


- ∇g has to be orthogonal to the constraint surface (red curve)
- At the optimum, ∇f and ∇g have to be parallel (in same or opposite direction)
- Hence, there must exist some $\lambda \in \mathbb{R}$ such that $\nabla f + \lambda \nabla g = 0$
- *Lagrangian function*: $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$
  λ is called *Lagrange multiplier*
- We obtain the solution to our optimization problem by setting both $\nabla_{\mathbf{x}} L = 0$ and $\frac{\partial L}{\partial \lambda} = 0$

Detour: Inequality constraints

- Suppose we want to find

$$\min_{\mathbf{w}} f(\mathbf{w})$$
$$\text{such that } g(\mathbf{w}) \geq 0$$



- In the interior ($g(\mathbf{x} > 0)$) - simply find $\nabla f(\mathbf{x}) = 0$
- On the boundary ($g(\mathbf{x} = 0)$) - same situation as before, but the sign matters this time
  For minimization, we want ∇f pointing in the same direction as ∇g

Detour: KKT conditions

- Based on the previous observations, let the Lagrangian be $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$
- We minimize L wrt $\mathbf{x}$ subject to the following constraints:

$$\begin{aligned} \lambda &\geq 0 \\ g(\mathbf{x}) &\geq 0 \\ \lambda g(\mathbf{x}) &= 0 \end{aligned}$$

- These are called *Karush-Kuhn-Tucker (KKT) conditions*

L_2 Regularization for linear models revisited

- Optimization problem: minimize error while keeping norm of the weights bounded

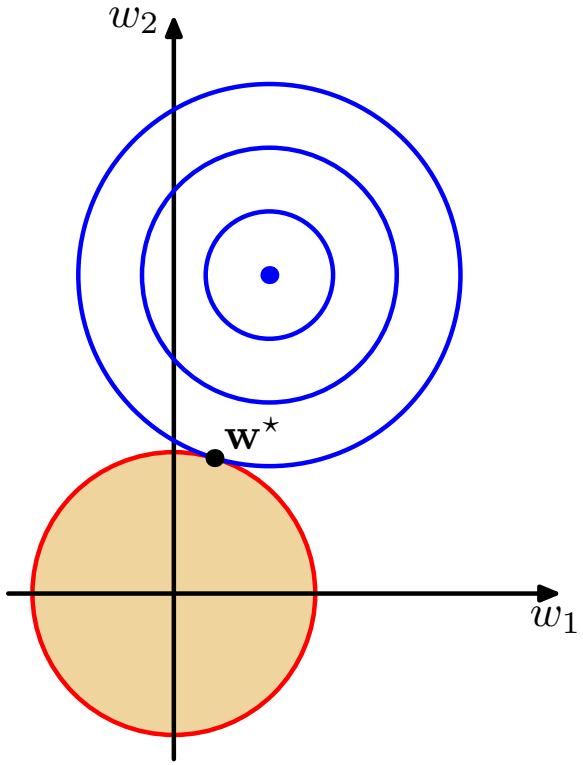
$$\begin{aligned} \min_{\mathbf{w}} J_D(\mathbf{w}) &= \min_{\mathbf{w}} (\mathbf{\Phi w} - \mathbf{y})^T (\mathbf{\Phi w} - \mathbf{y}) \\ \text{such that } \mathbf{w}^T \mathbf{w} &\leq \eta \end{aligned}$$

- The Lagrangian is:

$$L(\mathbf{w}, \lambda) = J_D(\mathbf{w}) - \lambda(\eta - \mathbf{w}^T \mathbf{w}) = (\mathbf{\Phi w} - \mathbf{y})^T (\mathbf{\Phi w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w} - \lambda \eta$$

- For a fixed λ, and $\eta = \lambda^{-1}$, the best $\mathbf{w}$ is the same as obtained by weight decay

Visualizing regularization (2 parameters)



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

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:

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

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

Solving L_1 regularization

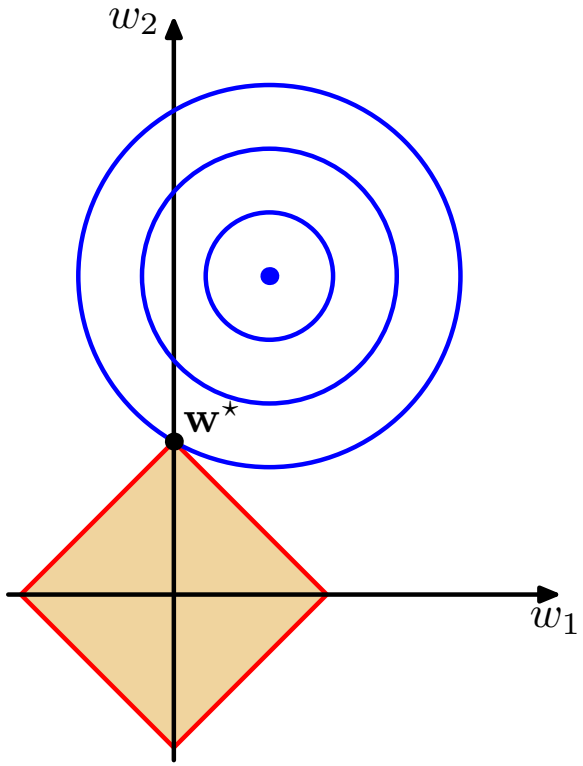
- The optimization problem is a quadratic program
- There is one constraint for each possible sign of the weights (2^n constraints for n weights)
- For example, with two weights:

$$\min_{w_1,w_2} \quad \sum_{j=1}^{m}(y_j - w_1x_1 - w_2x_2)^2$$

$$\begin{aligned} \text{such that } w_1 + w_2 &\leq \eta \\ w_1 - w_2 &\leq \eta \\ -w_1 + w_2 &\leq \eta \\ -w_1 - w_2 &\leq \eta \end{aligned}$$

- 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

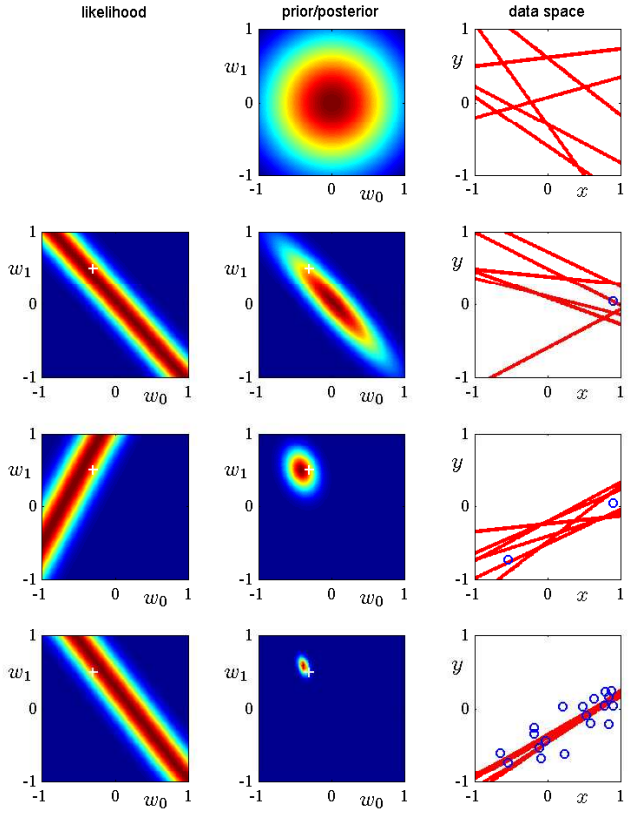
Pros and cons of L_1 regularization

- 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 at the moment

Bayesian view of regularization

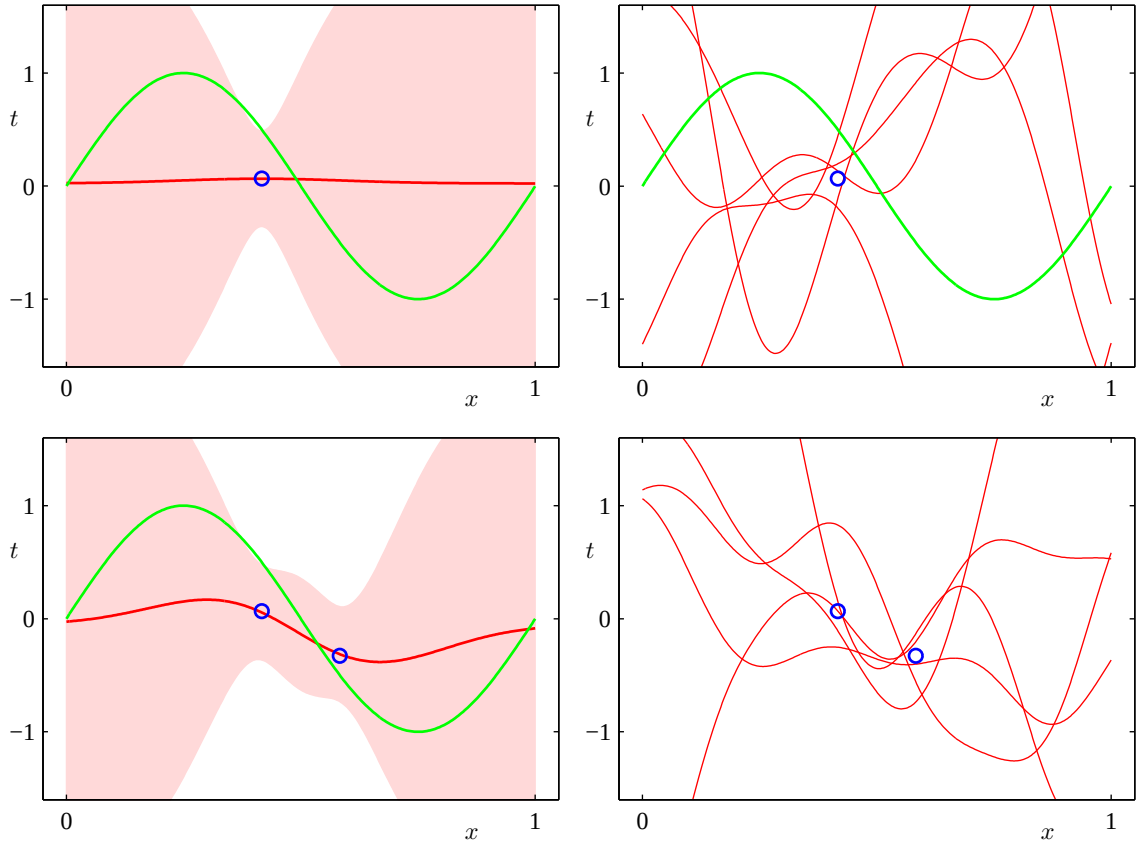
- Start with a *prior distribution* over hypotheses
- As data comes in, compute a *posterior distribution*
- We often work with *conjugate priors*, which means that when combining the prior with the likelihood of the data, one obtains the posterior in the same form as the prior
- Regularization can be obtained from particular types of prior (usually, priors that put more probability on simple hypotheses)
- E.g. L_2 regularization can be obtained using a circular Gaussian prior for the weights, and the posterior will also be Gaussian
- E.g. L_1 regularization uses double-exponential prior (see (Tibshirani, 1996))

Bayesian view of regularization



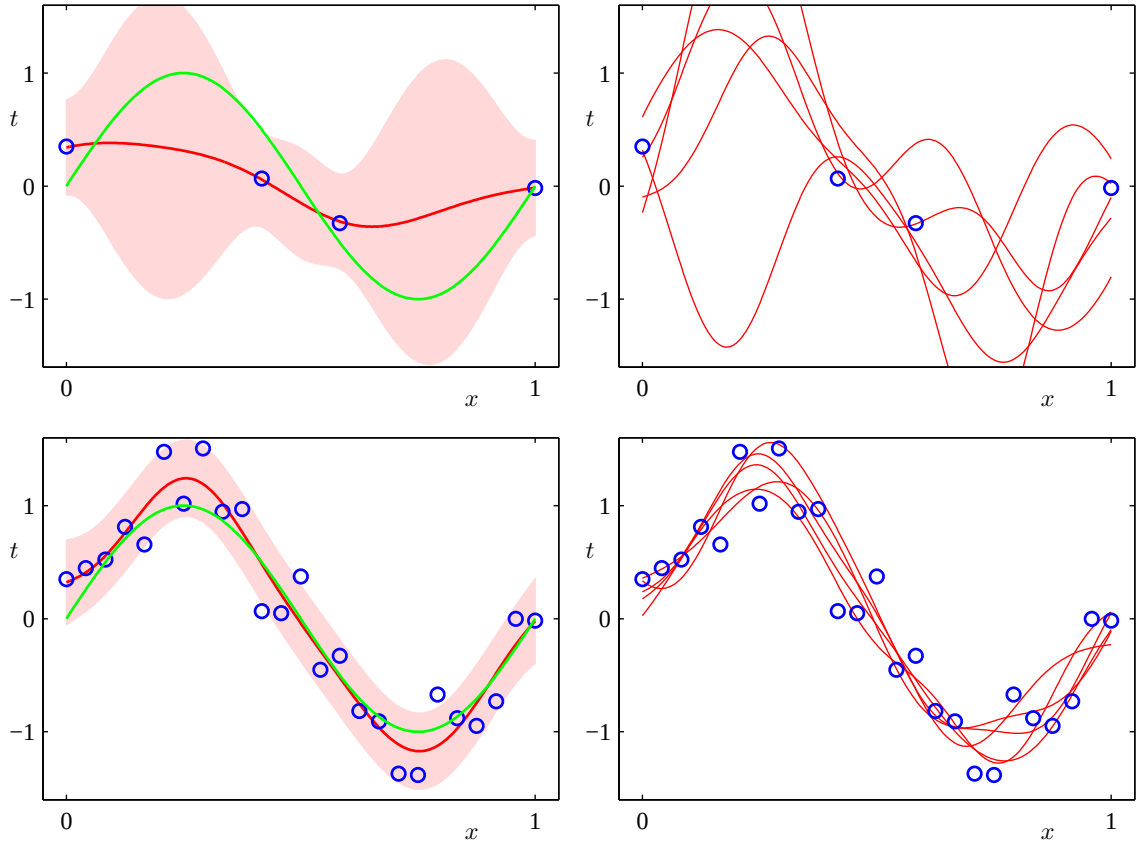
- Prior is round Gaussian
- Posterior will be skewed by the data

What does the Bayesian view give us?



- Circles are data points
- Green is the true function
- Red lines on right are drawn from the posterior distribution

What does the Bayesian view give us?



- Functions drawn from the posterior can be very different
- Uncertainty decreases where there are data points

What does the Bayesian view give us?

- Uncertainty estimates, i.e. how sure we are of the value of the function
- These can be used to guide active learning: ask about inputs for which the uncertainty in the value of the function is very high
- In the limit, Bayesian and maximum likelihood learning converge to the same answer
- In the short term, one needs a good prior to get good estimates of the parameters
- Sometimes the prior is overwhelmed by the data likelihood too early.
- Using the Bayesian approach does NOT eliminate the need to do cross-validation in general
- More on this later...