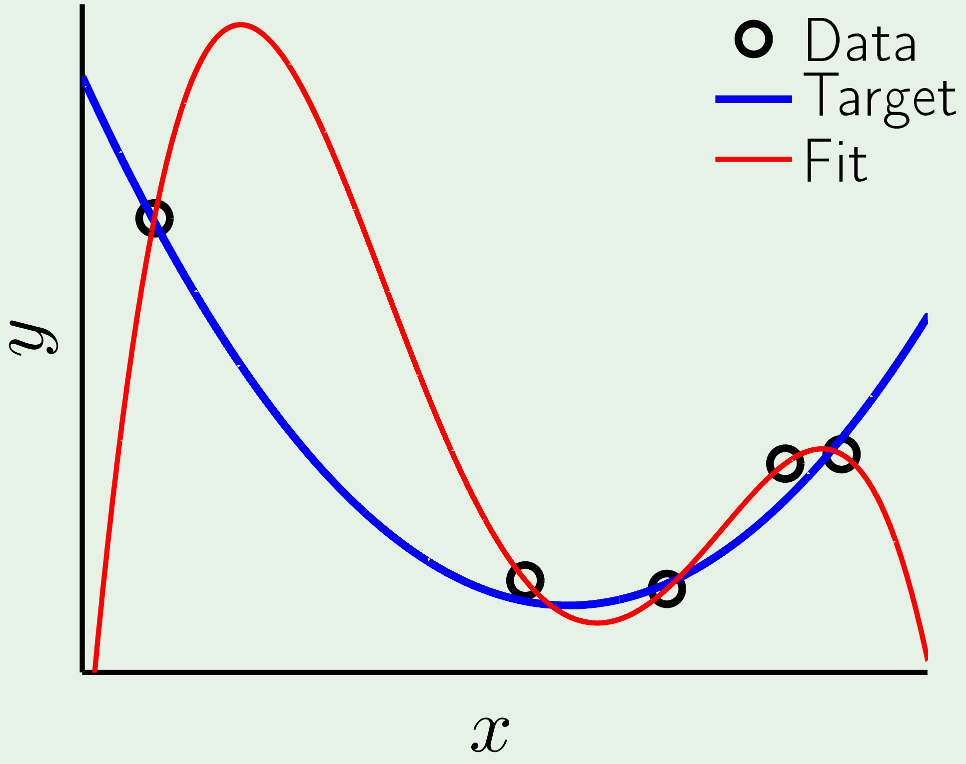
The source of overfitting is

Review of Lecture 11

- Overfitting

Fitting the data more than is warranted



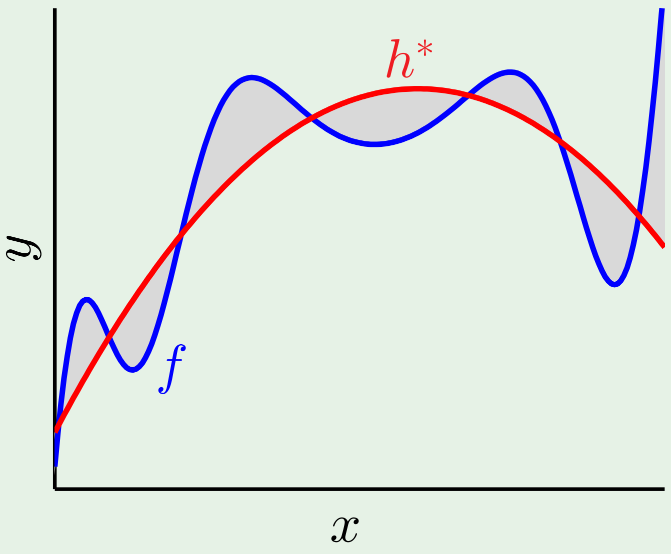
VC allows it; doesn't predict it

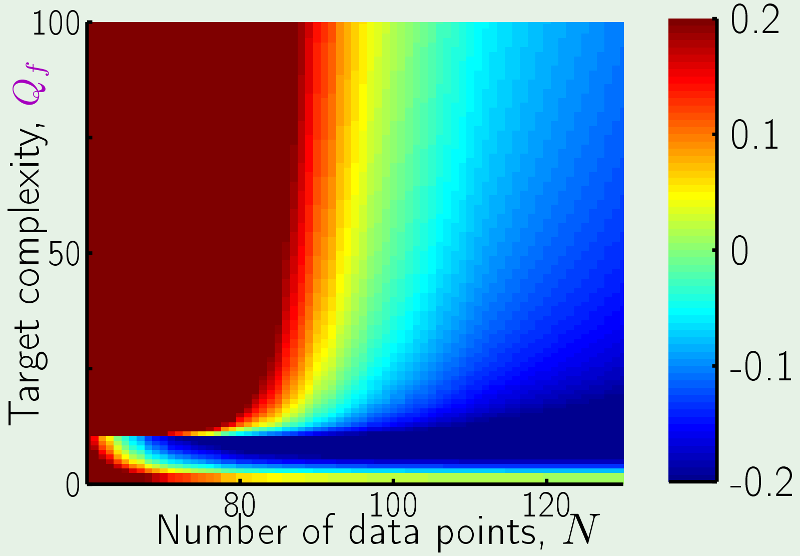
(as it gives a bound on the generalization error valid for all targets, noisy or noiseless)

Fitting the noise, stochastic/deterministic

Fitting the noise involves fitting something which cannot be fit, so we extrapolate out of sample to a non-existant pattern, and this pattern takes us away from the target function, so it will worsen Eout

- Deterministic noise





Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 12: Regularization

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Thursday, May 10, 2012

Outline

- Regularization - informal
- Regularization - formal
- Weight decay
- Choosing a regularizer

Two approaches to regularization

Mathematical:

Ill-posed problems in function approximation

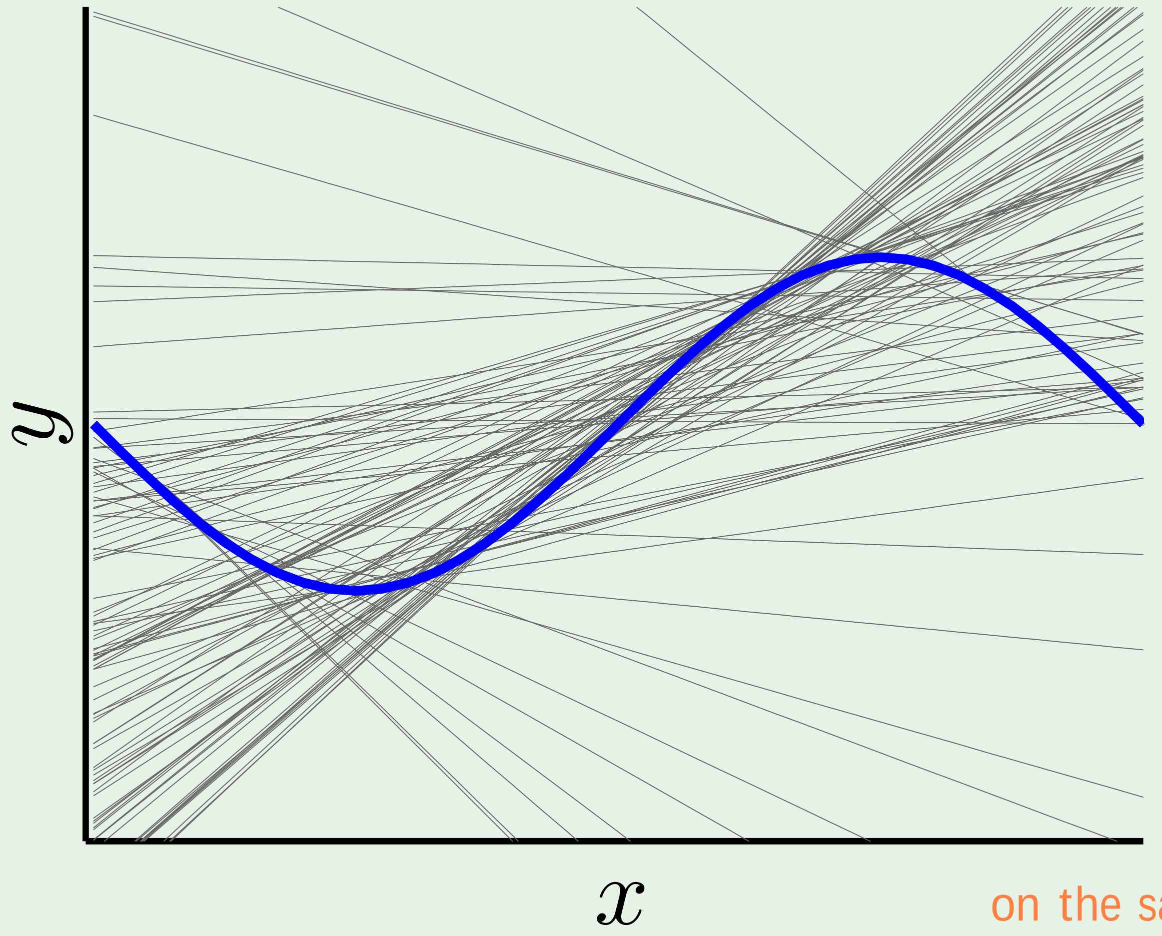
Ill-posed since several types of function can be used to approximate the function - so we impose smoothness constraints in order to solve it.

Heuristic:

Handicapping the minimization of E_{in}

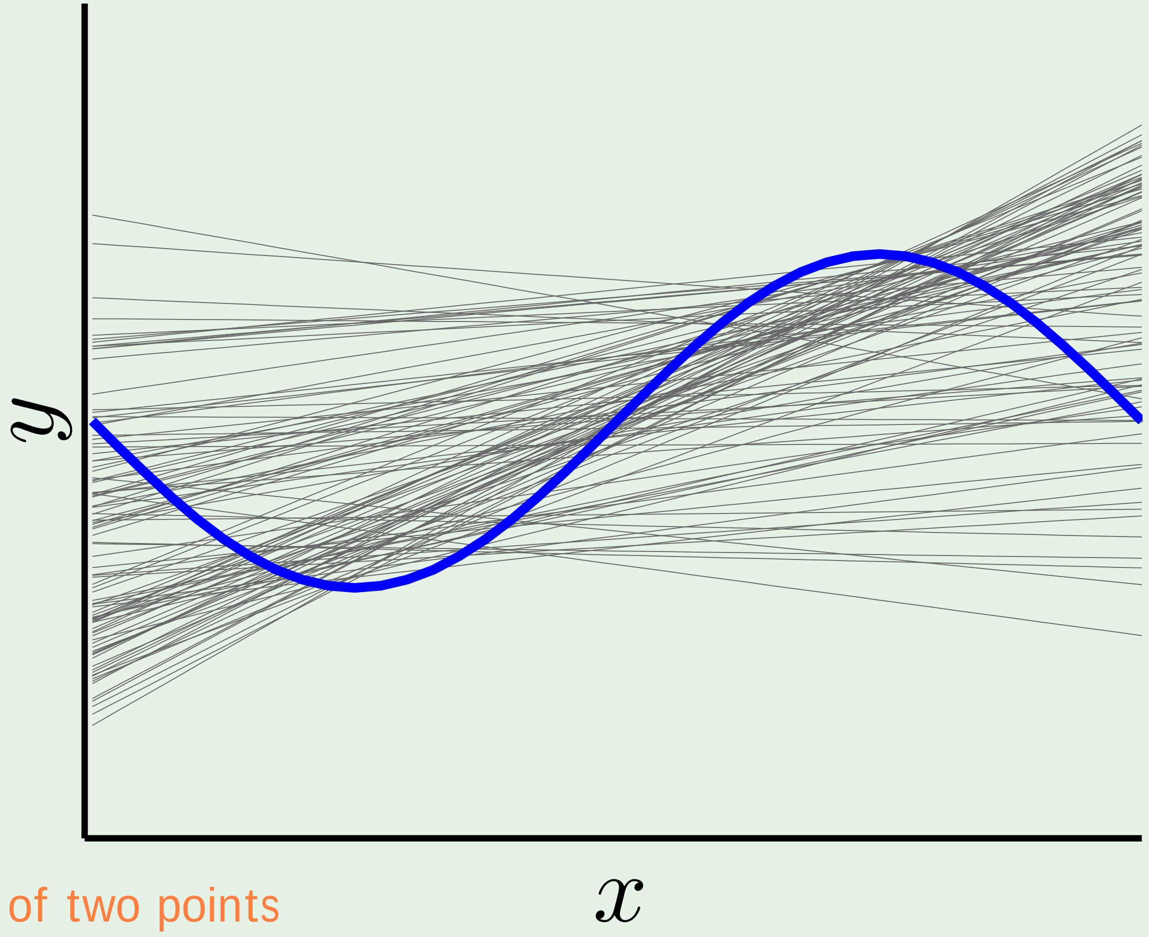
"Putting on the brakes in the algorithm"

A familiar example



without regularization

on the same sets of two points

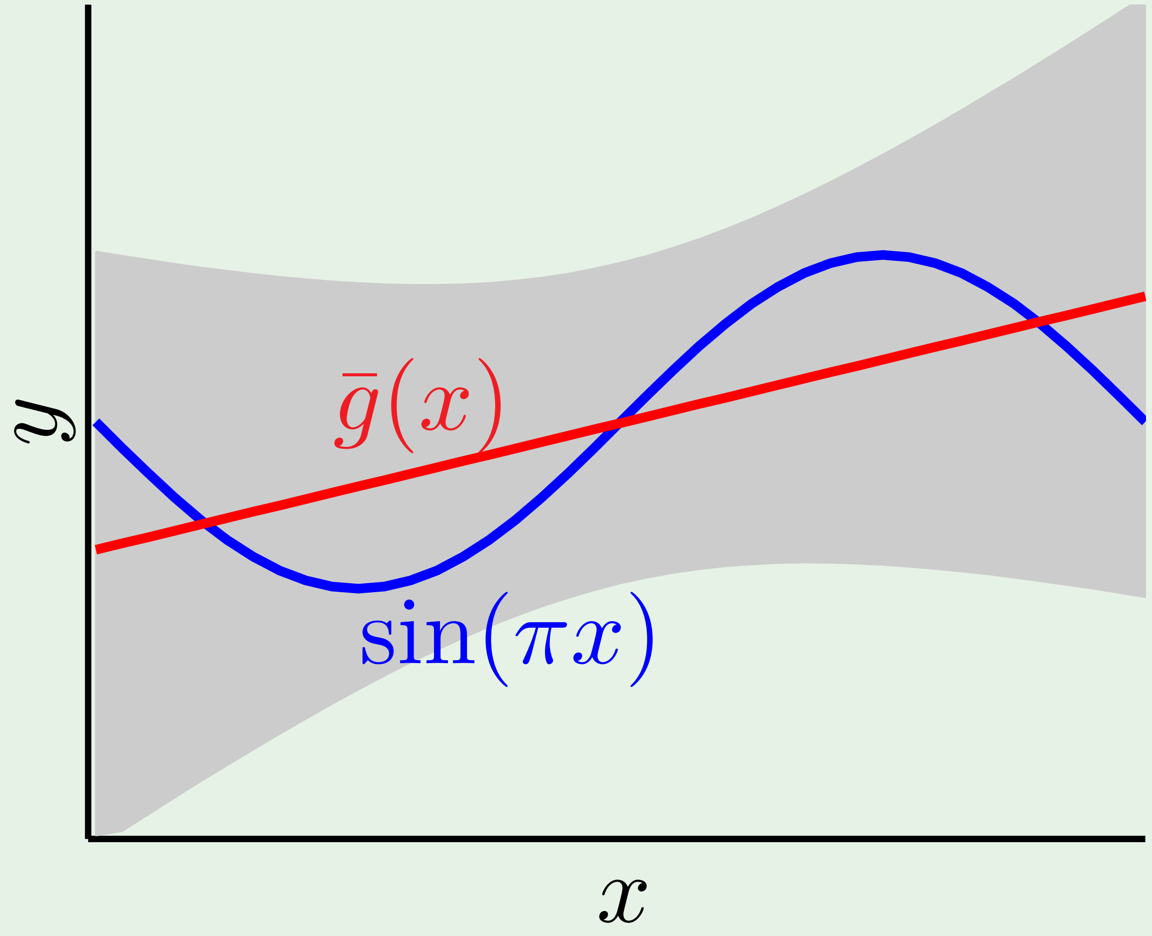


with regularization

we constrain the slope and offset of the lines:
it cannot pass through all the sets of points
perfectly (so bias increases) but the variance is reduced

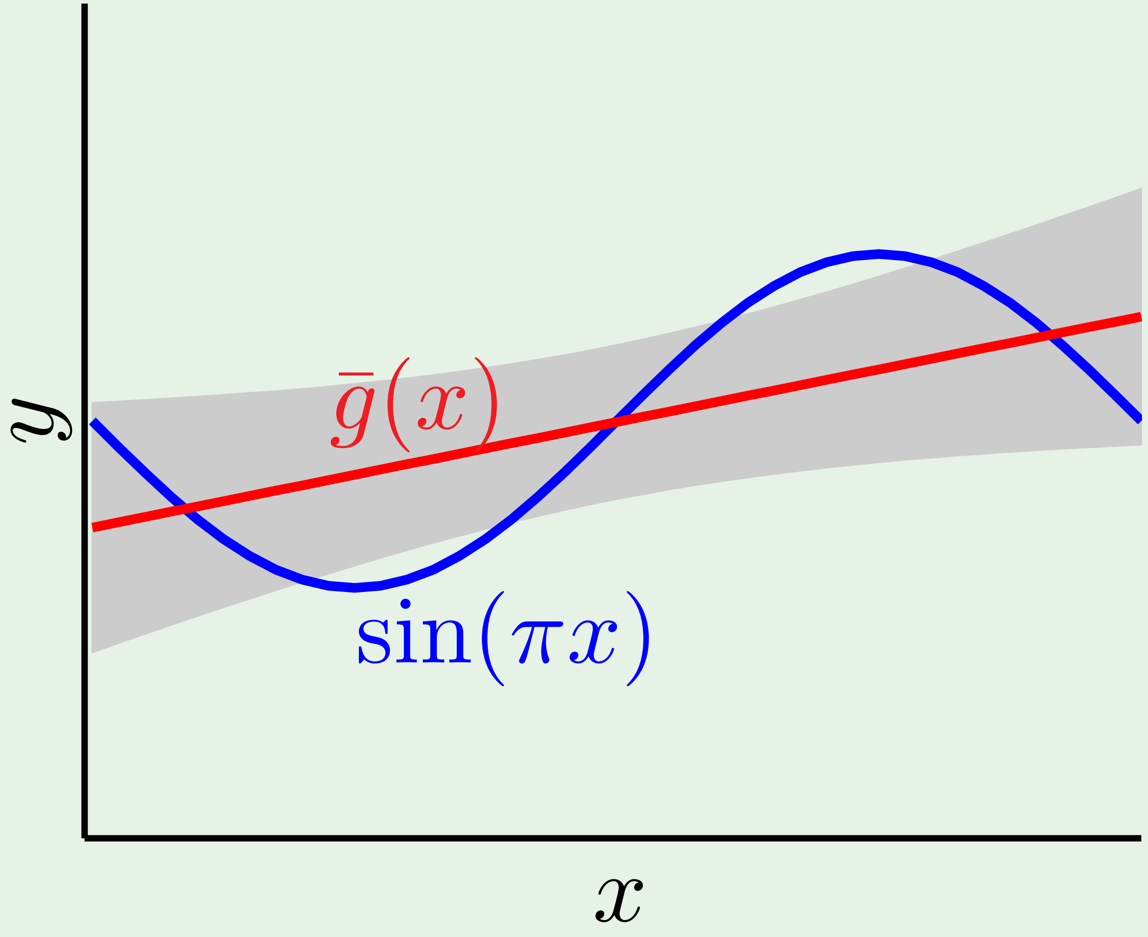
and the winner is ...

without regularization



bias $=$ **0.21** var $=$ **1.69**

with regularization



bias $=$ **0.23** var $=$ **0.33**

So we handicap the fit on both the signal and the noise together, but the handicapping of the noise is significant (leading to large reduction in variance) while the handicapping on the fit only causes a small bias increase. Comparing the numbers, this regularized line fit also wins over the constant model that we used before. So regularization allows us to find an intermediate in between the discrete sets of models (i.e. a 2nd order polynomial is just a restricted 4th order polynomial where the coefficients of x^3 and x^4 are zero). Regularization gives us a continuous set of models between extremely restricted to extremely unrestricted, we can fill in this gap and find the sweet spot which gives us the best out of sample error. Regularization can allow us to afford a greater VC dimension of our model provided we do the proper regularization. If before you could not afford to go to the bigger H (unconstrained) because the generalization would be too bad, we can try go to it anyway and apply regularization to see if the generalization is acceptable.

The polynomial model

$\mathcal{H}_Q$: polynomials of order Q

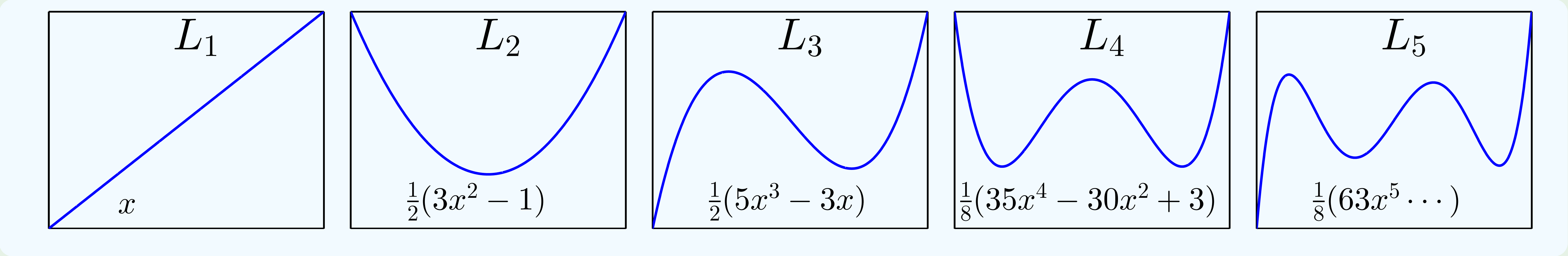
linear regression in $\mathcal{Z}$ space

Non-linear transformation vector z
takes scalar x and produces
the point in Z space.

$$\mathbf{z} = \begin{bmatrix} 1 \\ L_1(x) \\ \vdots \\ L_Q(x) \end{bmatrix} \qquad \mathcal{H}_Q = \left\{ \sum_{q=0}^{Q} w_q \, L_q(x) \right\}$$

Again, Legendre used because they are individually orthogonal to each other - so they can be used as form of basis set and combined with different coefficients. These coefficients will be independent of each other, they deal with different coordinates which do not interfere with eachother. Monomials are heavily correlated so the relevant parameter will be a combination of the weights as opposed to an individual weight - instead we get the combination ahead of time so the weights are meaningful in their own right.

Legendre polynomials:



Unconstrained solution

Given $(x_1, y_1), \cdots, (x_N, y_n) \quad \longrightarrow \quad (\mathbf{z}_1, y_1), \cdots, (\mathbf{z}_N, y_n)$

Minimize $E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^{\mathsf{T}} \mathbf{z}_n - y_n)^2$

Minimize $\frac{1}{N} (\mathrm{Z}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathrm{Z}\mathbf{w} - \mathbf{y})$

$$\mathbf{w}_{\text{lin}} = (\mathrm{Z}^{\mathsf{T}}\mathrm{Z})^{-1}\mathrm{Z}^{\mathsf{T}}\mathbf{y}$$

Constraining the weights

Hard constraint: $\mathcal{H}_2$ is constrained version of $\mathcal{H}_{10}$ with $w_q = 0$ for $q > 2$

Softer version: $\displaystyle\sum_{q=0}^{Q} w_q^2 \leq C$ "**soft-order**" constraint

C is the "budget" - this soft constraint still reduces the number of effective d.o.f./VC dimension, so generalization should improve

Minimize $\frac{1}{N}(\mathrm{Z}\mathbf{w} - \mathbf{y})^\intercal(\mathrm{Z}\mathbf{w} - \mathbf{y})$

subject to: $\mathbf{w}^\intercal\mathbf{w} \leq C$

Solution: $\mathbf{w}_{\mathrm{reg}}$ instead of $\mathbf{w}_{\mathrm{lin}}$

Solving for $\mathbf{w}_{\text{reg}}$

Minimize $\quad E_{\text{in}}(\mathbf{w}) \; = \; \frac{1}{N}\,(\mathrm{Z}\mathbf{w}-\mathbf{y})^{\mathsf{T}}(\mathrm{Z}\mathbf{w}-\mathbf{y})$

subject to: $\quad \mathbf{w}^{\mathsf{T}}\mathbf{w} \le C$

From the diagram, we require:

$$\nabla E_{\text{in}}(\mathbf{w}_{\text{reg}}) \propto \; -\mathbf{w}_{\text{reg}}$$

$$= \; -2\frac{\lambda}{N}\mathbf{w}_{\text{reg}}$$

$$\nabla E_{\text{in}}(\mathbf{w}_{\text{reg}}) + 2\frac{\lambda}{N}\mathbf{w}_{\text{reg}} \; = \; \mathbf{0}$$

When the vector w, which is normal to wTw=C, and grad(Ein) are opposite (so grad(Ein no longer has a component along the tangent to the red circle, so Ein can no longer decrease), then w is the minimum given the constraint.

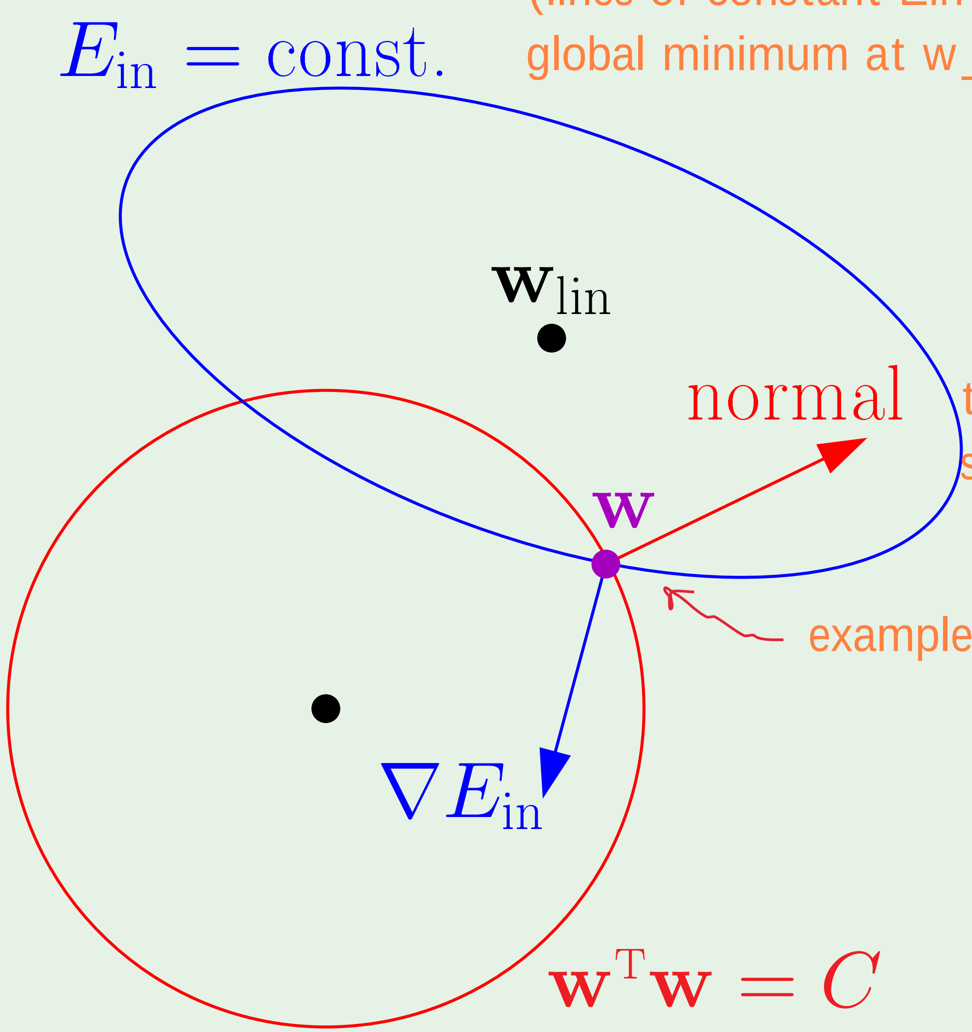
The above expression is from the minimization of:

Minimize $\quad E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^{\mathsf{T}}\mathbf{w}$

$$\boxed{C\uparrow \quad \lambda\downarrow}$$

For large C, w_lin is the solution and we should be minimizing Ein as if there were no constraint, so lambda = 0. With smaller C, the regularization (limit on wTw) is more severe, so the regularization term in the expression must have a bigger magnitude, so lambda increases. If C = 0, Ein is a single value, w = 0 and we have infinite lambda, so the expression is just Ein(0). Note we will use validation to decide the value of lambda in a principled way.

(lines of constant Ein in blue, global minimum at w_lin)



to the red surface

example point w

Given the constraint the minimum of Ein is when wTw = C

Augmented error

Minimizing $$E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^{\mathsf{T}}\mathbf{w}$$

$$= \frac{1}{N}\,(\mathrm{Z}\mathbf{w} - \mathbf{y})^{\mathsf{T}}(\mathrm{Z}\mathbf{w} - \mathbf{y}) + \frac{\lambda}{N}\mathbf{w}^{\mathsf{T}}\mathbf{w}$$ unconditionally

(unconstrained optimization)

– solves –

Minimizing $$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\,(\mathrm{Z}\mathbf{w} - \mathbf{y})^{\mathsf{T}}(\mathrm{Z}\mathbf{w} - \mathbf{y})$$

subject to: $\mathbf{w}^{\mathsf{T}}\mathbf{w} \le C$ $\longleftarrow$ VC formulation

This formulation lends itself to VC analysis, since we are restricting the hypothesis set explicitly (we are using a subset of H) and we expect good generalization. The augmented error formulation (i.e. the unconstrained version) does not prohibit any h but we have a preference of weights based on a penalty (which is related to the constraint) - it uses a different learning algorithm to find the solution w (which will be a member of H_Q).

The solution

Minimize $\quad E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^{\mathsf{T}}\mathbf{w}$

$$= \frac{1}{N}\left((\mathrm{Z}\mathbf{w}-\mathbf{y})^{\mathsf{T}}(\mathrm{Z}\mathbf{w}-\mathbf{y}) \ + \ \lambda\ \mathbf{w}^{\mathsf{T}}\mathbf{w}\right)$$

$\nabla E_{\text{aug}}(\mathbf{w}) = \mathbf{0} \quad \Longrightarrow \quad \mathrm{Z}^{\mathsf{T}}(\mathrm{Z}\mathbf{w}-\mathbf{y}) + \lambda\mathbf{w} = \mathbf{0}$

$$\boxed{\mathbf{w}_{\text{reg}} = (\mathrm{Z}^{\mathsf{T}}\mathrm{Z}+\lambda\mathrm{I})^{-1}\,\mathrm{Z}^{\mathsf{T}}\mathbf{y}}$$ (with regularization)

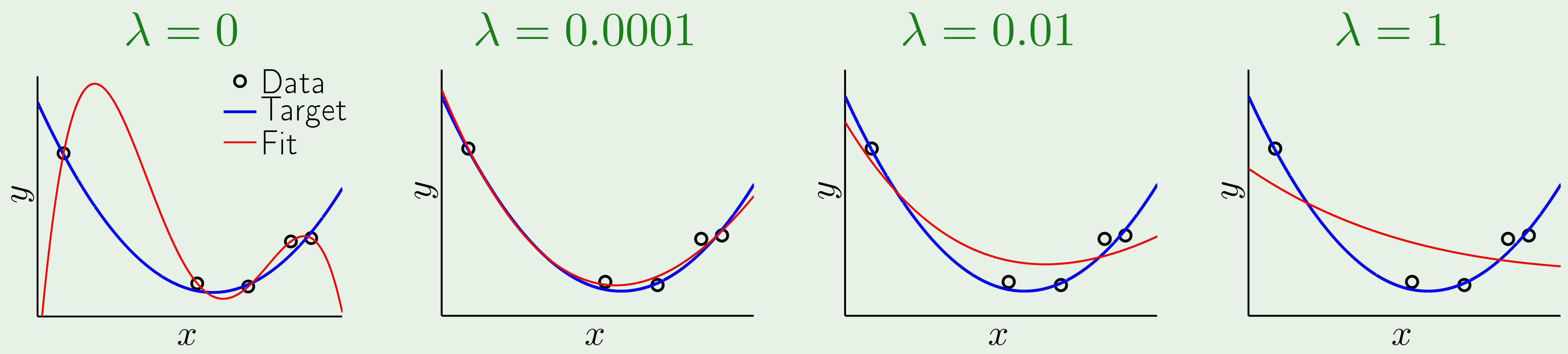
(still one-step learning with regularization)

as opposed to $\quad \mathbf{w}_{\text{lin}} = (\mathrm{Z}^{\mathsf{T}}\mathrm{Z})^{-1}\mathrm{Z}^{\mathsf{T}}\mathbf{y}$ (without regularization)

Increasing lambda means the regularization term becomes dominant in the solution. If lambda is enormous, its term completely dominates, so the inversion tends to the inverse of lambda, which is very small, w_reg is very close to/tends towards zero (which is equivalent to the case when C=0 from before).

The result

Minimizing $\quad E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\,\mathbf{w}^{\intercal}\mathbf{w} \quad$ for different λ's:



overfitting $\longrightarrow$ $\longrightarrow$ $\longrightarrow$ $\longrightarrow$ underfitting

Weight 'decay'

(form of regularizer)

Minimizing $\quad E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\,\mathbf{w}^{\mathsf{T}}\mathbf{w} \quad$ is called weight *decay*. Why?

(Batch)

Gradient descent:

$$\mathbf{w}(t+1) = \mathbf{w}(t) \;-\; \eta\,\nabla E_{\text{in}}\big(\mathbf{w}(t)\big) \;-\; 2\,\eta\,\frac{\lambda}{N}\,\mathbf{w}(t)$$

$$= \mathbf{w}(t)\,\left(1 - 2\eta\frac{\lambda}{N}\right) - \eta\,\nabla E_{\text{in}}\big(\mathbf{w}(t)\big)$$

So the weights are shrunk (in magnitude) towards the origin, then moved along negative grad(Ein). Hence weight decay, since the weight decays toward the origin from one iteration to the next. For very large lambda, the decay factor is large so the weight will soon be at the origin and there is no learning of the function itself (since the learning factor is small compared to lambda).

Applies in neural networks:

$$\mathbf{w}^{\mathsf{T}}\mathbf{w} = \sum_{l=1}^{L}\sum_{i=0}^{d^{(l-1)}}\sum_{j=1}^{d^{(l)}}\left(w_{ij}^{(l)}\right)^2$$

since a weight exists in the link between each connected neuron in the network.

Variations of weight decay

Instead of a uniform budget C,

Emphasis of certain weights: $\sum_{q=0}^{Q} \gamma_q \, w_q^2$

importance factor gamma

Examples:

$\gamma_q = 2^q \quad \Longrightarrow$ **low-order fit**

more emphasis in the constraint on higher order terms, so trying to find as much as possible a low-order fit

$\gamma_q = 2^{-q} \quad \Longrightarrow$ **high-order fit**

Neural networks: different layers get different γ's

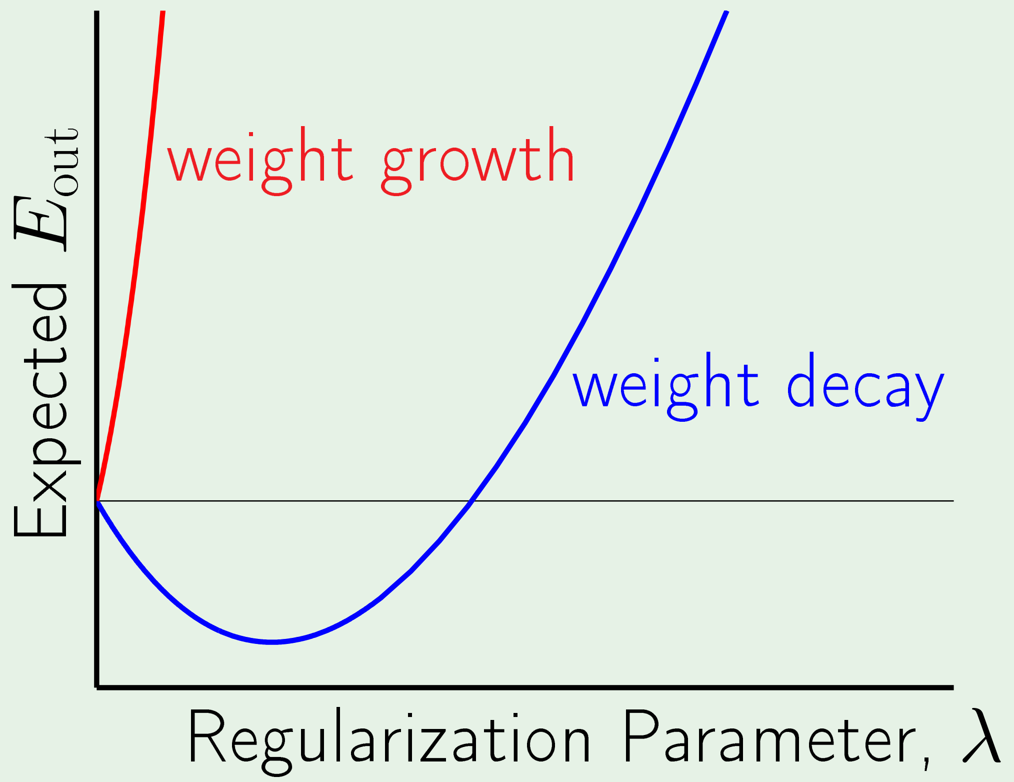
Doing the analysis properly for neural networks, you find the best way to do weight decay it to give different emphasis on the weights of different layers since they play a different role on affecting the output. So this is accommodated for by having different gamma for different layers. While the above form is diagonal quadratic (i.e. only take w1^2, w2^2 etc.), the Tikhonov regularizer (in matrix form) is a general quadratic form so has diagonals and off-diagonals. Therefore it gives weights to w1*w3 etc. This means with a proper choice of matrix gamma, you can have weight decay, low-order, high-order, and other variations. This general form is therefore very interesting as you can cover a lot of territory using it.

Tikhonov regularizer: $\mathbf{w}^{\intercal}\Gamma^{\intercal}\Gamma\mathbf{w}$

Even weight growth!

We 'constrain' the weights to be large - bad!

We need to choose a regularizer to prevent overfitting as it is a cure for fitting the noise. There are practical rules for choosing (see below) and after we choose there is the check of the lambda. If we choose the wrong one, the correct validation will recommend that lambda=0.



Practical rule:

Practical observations:

stochastic noise is 'high-frequency'

is not as high-freq, but

deterministic noise is also non-smooth

We capture what we can with the model, and anything we can't capture is likely going up and down stronger/faster than we can capture

Rule: $\Longrightarrow$ constrain learning towards smoother hypotheses

We want to punish the noise more than we punish the signal, so a regularizer which prefers smooth hypotheses will more likely fail to fit the noise rather than the signal. In most of the parameterizations of hypothesis sets, smaller weights correspond to smoother hypotheses - so weight decay works well in these cases.

General form of augmented error

Calling the regularizer $\Omega = \Omega(h)$, we minimize

$$E_{\text{aug}}(h) = E_{\text{in}}(h) + \frac{\lambda}{N}\Omega(h)$$

Rings a bell? $\qquad \downarrow \downarrow$

$$E_{\text{out}}(h) \leq E_{\text{in}}(h) + \Omega(\mathcal{H})$$

E_{aug} is better than E_{in} as a proxy for E_{out}

You can think of the holy grail of machine learning as finding an in sample estimate of the out of sample error. If you get that, you are done, minimize it and you go home. Here, Eaug is a better proxy for Eout than Ein.

Outline

- Regularization - informal
- Regularization - formal
- Weight decay
- Choosing a regularizer

If you identify two regularizers that behave differently in different parts of the space, it can be useful to have a combination of them for the same learning problem.

If the (heuristic) choice of the regularizer (from studying the problem) is not that great, the saving grace will be the principled determination of lambda via validation.

The perfect regularizer Ω

Constraint in the 'direction' of the target function (going in circles ☺)

Regularization simply reduces overfitting by applying generically a methodology which harms the overfitting more than it harms the fitting (i.e it harms fitting the noise more than it harms fitting the signal). Hence it is a heuristic.

Guiding principle:

Direction of **smoother** or "simpler"

since the stochastic and deterministic noise is not smooth (they are high-frequency, in the latter case relative to the hypothesis set)

Chose a bad Ω?

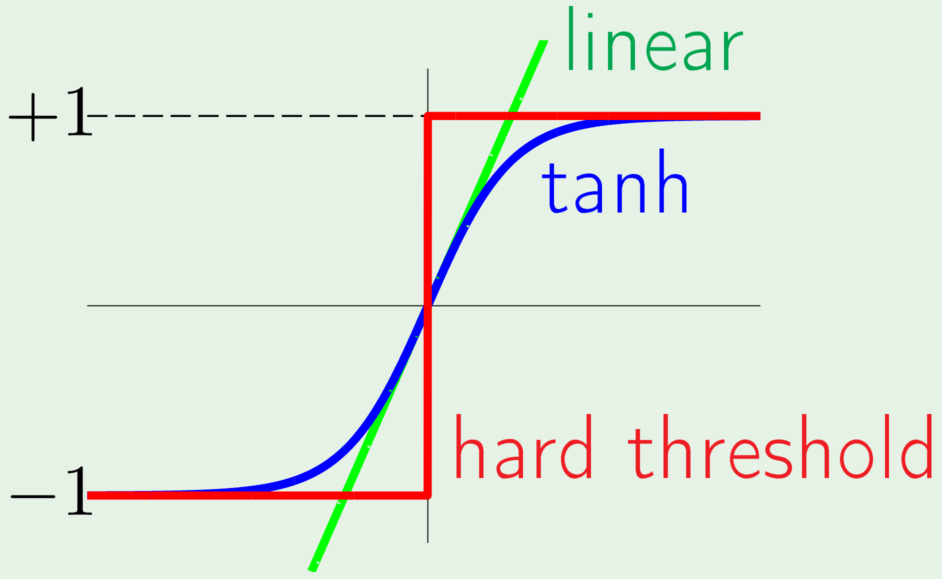
We still have λ!

In the movie rating example there is no clear concept of "smoother". Here, the best-found regularization pulls the solution/weights towards giving the average rating (most likely the average of all the movies the user has seen) which is the "simpler" solution.

Neural-network regularizers

Weight decay: From linear to logical

If the weights are small, the signal will be linear - in a multilayer network, it can still only produce/ implement a linear function. Increasing the weights gives us more non-linearity in the signal, up to a limit of logical dependency which can implement any functionality you want.



Weight elimination:

If VC dimension is approximately the number of weights,

Fewer weights $\Longrightarrow$ smaller VC dimension

so better chance of generalizing and perhaps we will not overfit

Soft weight elimination:

$$\Omega(\mathbf{w}) = \sum_{i,j,l} \frac{\left(w_{ij}^{(l)}\right)^2}{\beta^2 + \left(w_{ij}^{(l)}\right)^2}$$

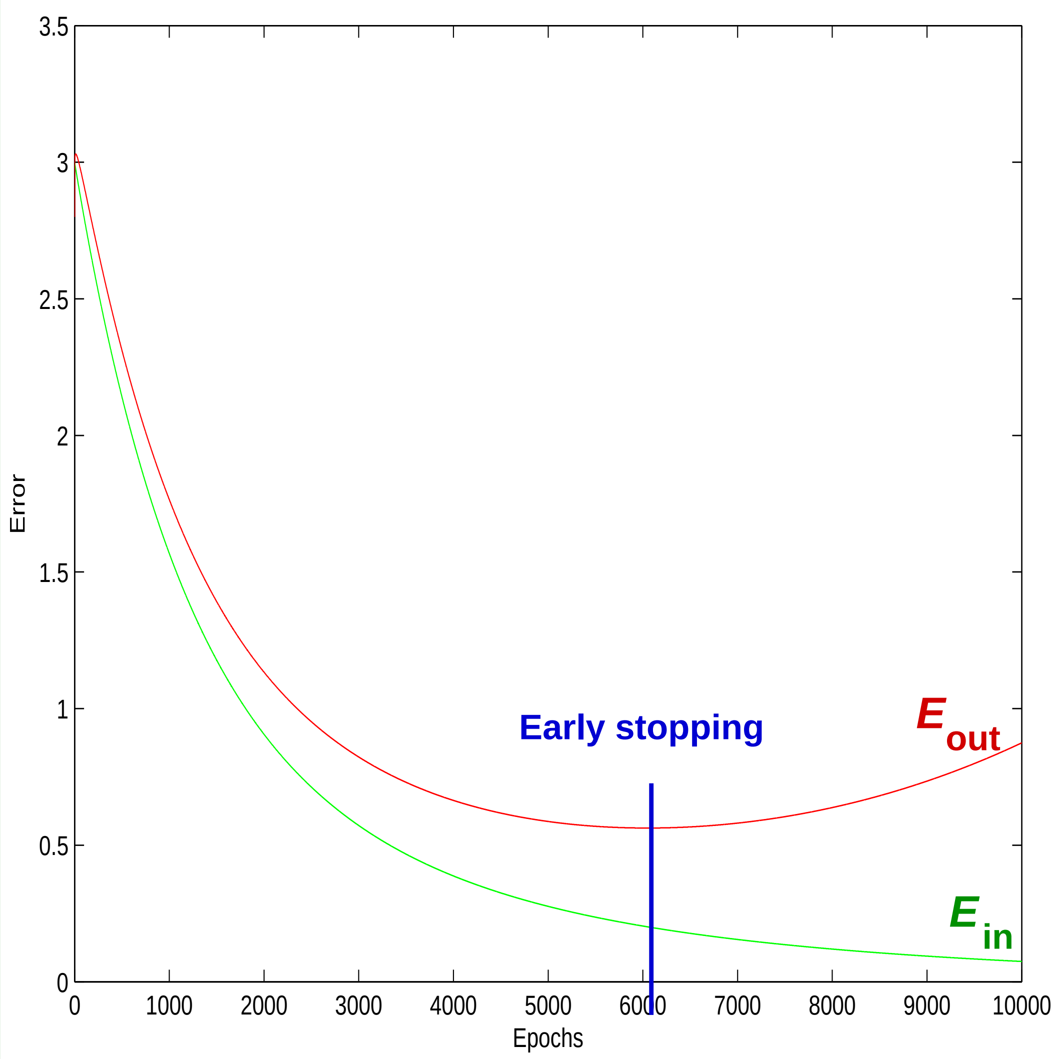
For very small w, beta dominates, so the expression is proportional to w^2, so we are doing weight decay. For very large w, the expression is 1, so not much to be gained from changing the weights. Hence big weights are left alone, while small weights are pushed towards zero. We end up after the optimization with two groups: serious weights and weights which are being pushed towards zero which are considered to be (soft) eliminated.

Early stopping as a regularizer

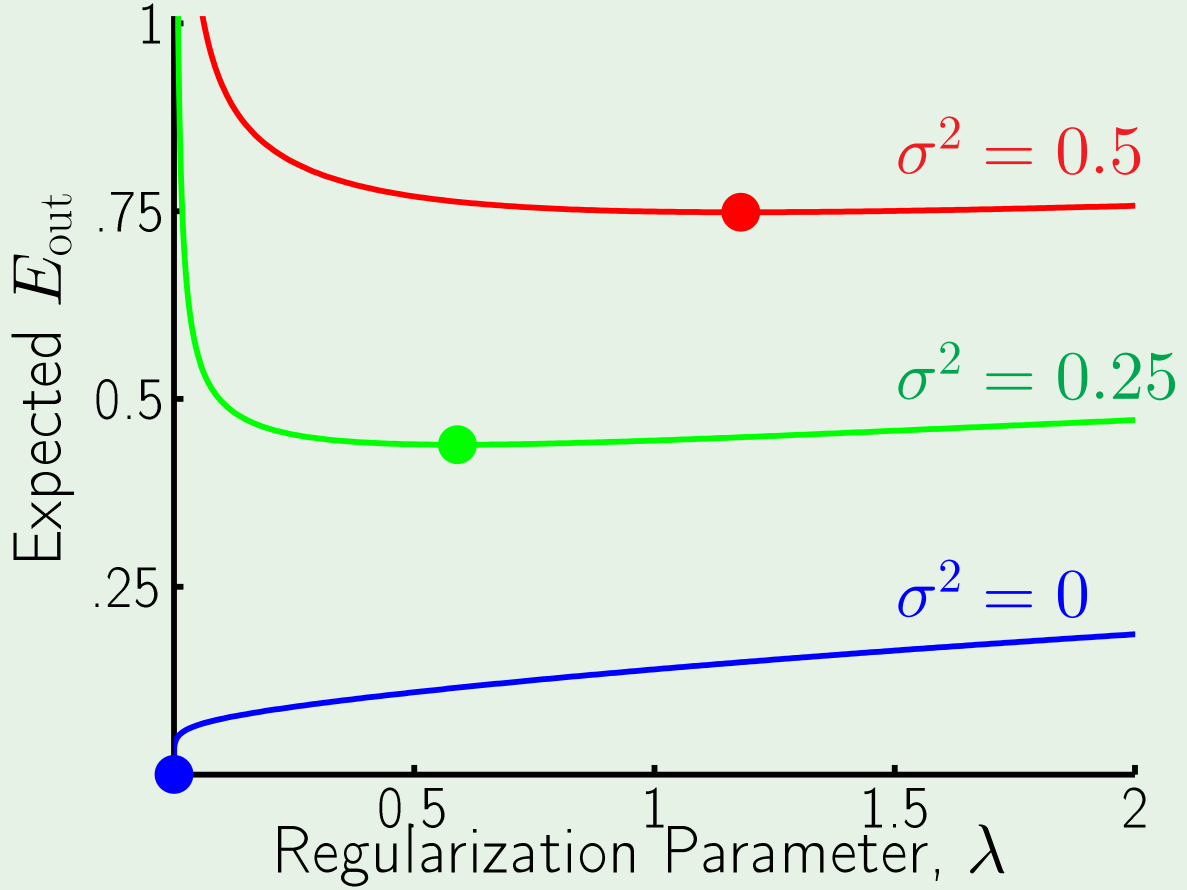
Regularization through the optimizer!

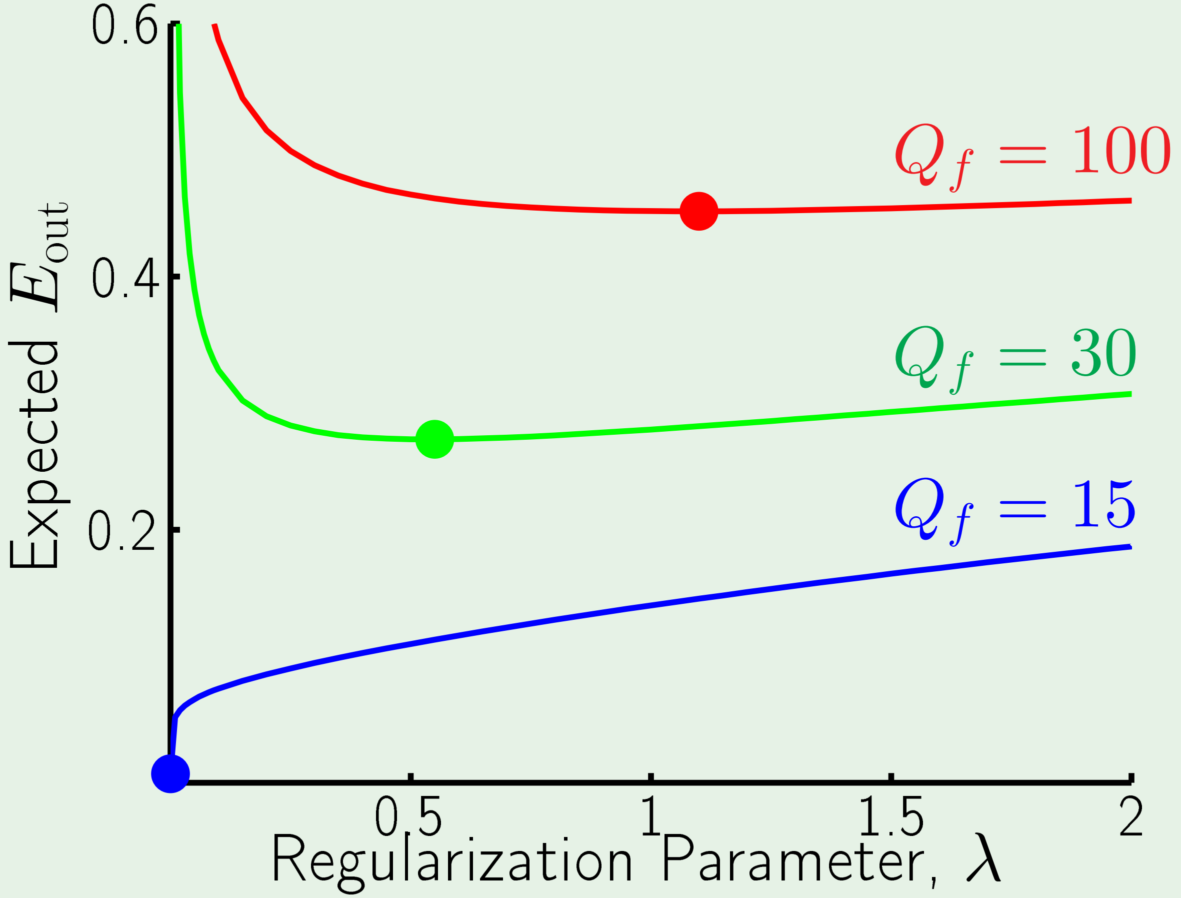
When to stop? **validation**

It is best to choose a regularizer and put it in the augmented error function, then give it to the optimizer to go all the way in minimizing. This is better than hoping the optimizer gets stuck in a local minimum (i.e. that it does a poor job of optimizing) and hoping that this is sufficient regularization to reduce overfitting. We want to capture as much as possible in the objective function (the error function we are minimizing) and we know that we really want to minimize it, then we have a principled way of doing that and we get what we want.



The optimal λ





Stochastic noise **Deterministic noise**

With regards to the correspondence between the two types of noise: as far as overfitting, and its cures, are concerned: deterministic noise behaves almost exactly as if it were unknown stochastic noise.