Applied Machine Learning

Regularization

Reihaneh Rabbany

McGill
School of Computer Science

Learning objectives

- intuition for model complexity and overfitting
- regularization penalty (L1 & L2)
- probabilistic interpretation
- bias and variance trade-off

recall

Linear regression

model

$$f_w(x) = \hat{y}^{(n)} = w^T x^{(n)}$$

$\in \mathbb{R}$ $\quad 1 \times D \quad D \times 1$

cost

$$J(w) = \frac{1}{2} \sum_n \left(y^{(n)} - w^T x^{(n)} \right)^2$$

linear least squares **(LLS)**

Optimization

$$\sum_n (y^{(n)} - w^T x^{(n)}) x_d^{(n)} = 0 \quad \forall d$$

what if **linear fit is not the best**?
use nonlinear basis

matrix notation

$$\hat{y} = Xw$$

$N \times 1 \quad N \times D \quad D \times 1$

$$J(w) = \frac{1}{2} ||y - Xw||^2$$
$$= \frac{1}{2} (y - Xw)^T (y - Xw)$$

$$X^T (y - Xw) = \vec{0}$$

$$w^* = (X^T X)^{-1} X^T y$$

$D \times 1 \quad D \times N \quad N \times D \quad N \times 1$

Previously...

Linear regression and logistic regression

is linear too simple? what if it's not a good fit?

how to increase the model's expressiveness?

- include new features from the domain
- create new nonlinear features from the existing ones

recall

Nonlinear basis functions

replace original features in $f_w(x) = \sum_d w_d x_d$

with nonlinear bases $f_w(x) = \sum_d w_d \phi_d(x)$

linear least squares solution $(\Phi^\top \Phi) w^* = \Phi^\top y$

replacing X with Φ

a (nonlinear) feature

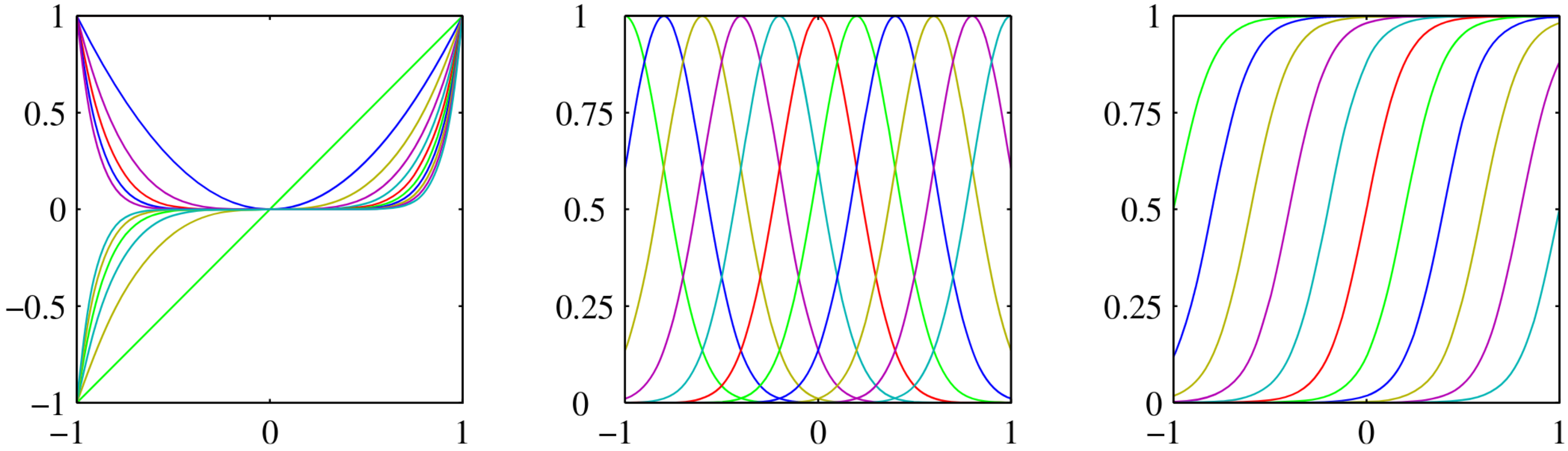
$$\Phi = \begin{bmatrix} \phi_1(x^{(1)}), & \phi_2(x^{(1)}), & \cdots, & \phi_D(x^{(1)}) \\ \phi_1(x^{(2)}), & \phi_2(x^{(2)}), & \cdots, & \phi_D(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x^{(N)}), & \phi_2(x^{(N)}), & \cdots, & \phi_D(x^{(N)}) \end{bmatrix}$$

one instance

recall

Nonlinear basis functions

examples original input is scalar $x \in \mathbb{R}$



polynomial bases

$$\phi_k(x) = x^k$$

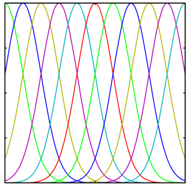
Gaussian bases

$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$

Sigmoid bases

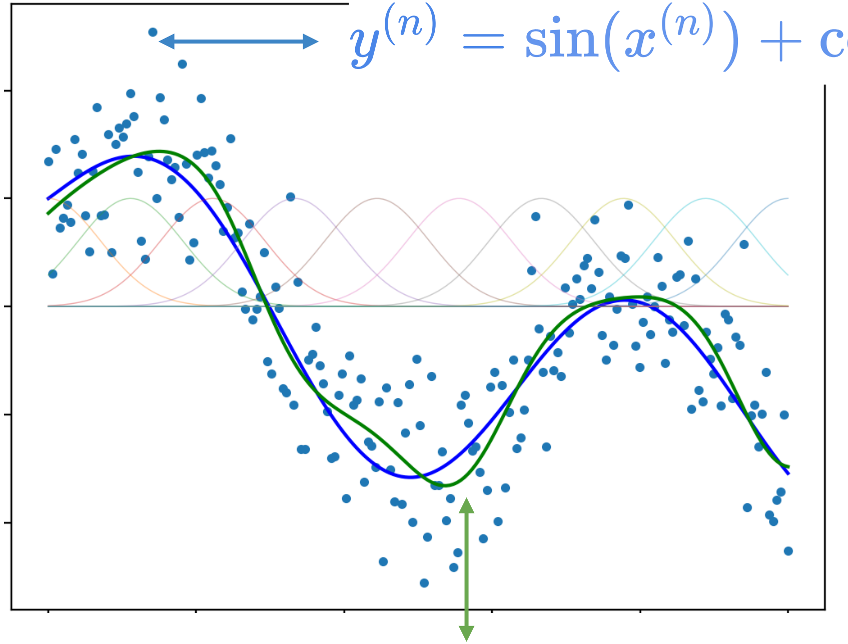
$$\phi_k(x) = \frac{1}{1+e^{-\frac{x-\mu_k}{s}}}$$

Example: Gaussian bases



$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$

$y^{(n)} = \sin(x^{(n)}) + \cos(\sqrt{|x^{(n)}|}) + \epsilon$



our fit to data using 10 Gaussian bases

prediction for a new instance

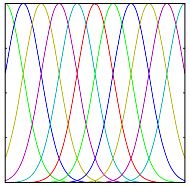
$$f(x') = \phi(x')^\top (\Phi^\top \Phi)^{-1} \Phi^\top y$$

new instance

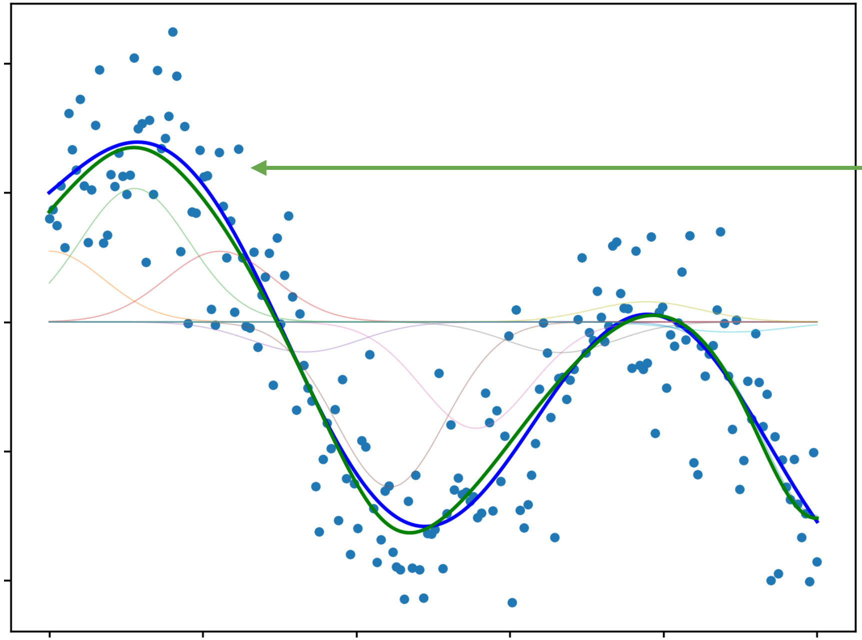
features evaluated for the new point

w found using LLS

Example: Gaussian bases



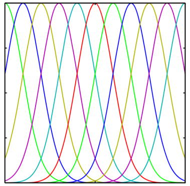
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



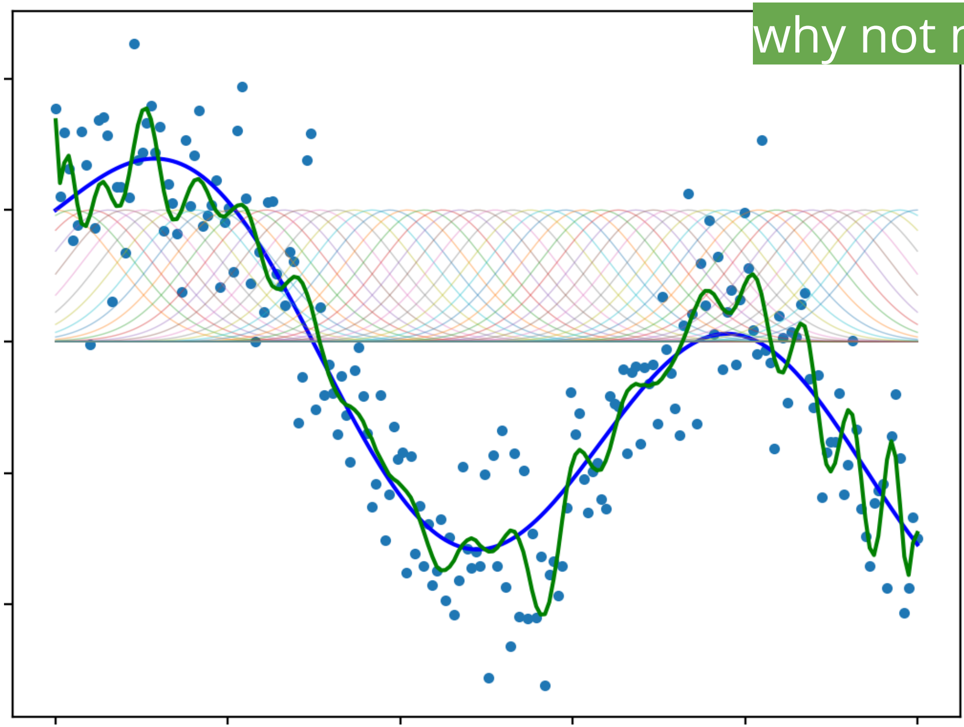
our fit to data using **10 Gaussian bases**

why not more?

Example: Gaussian bases



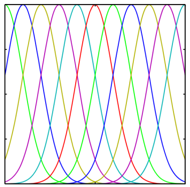
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



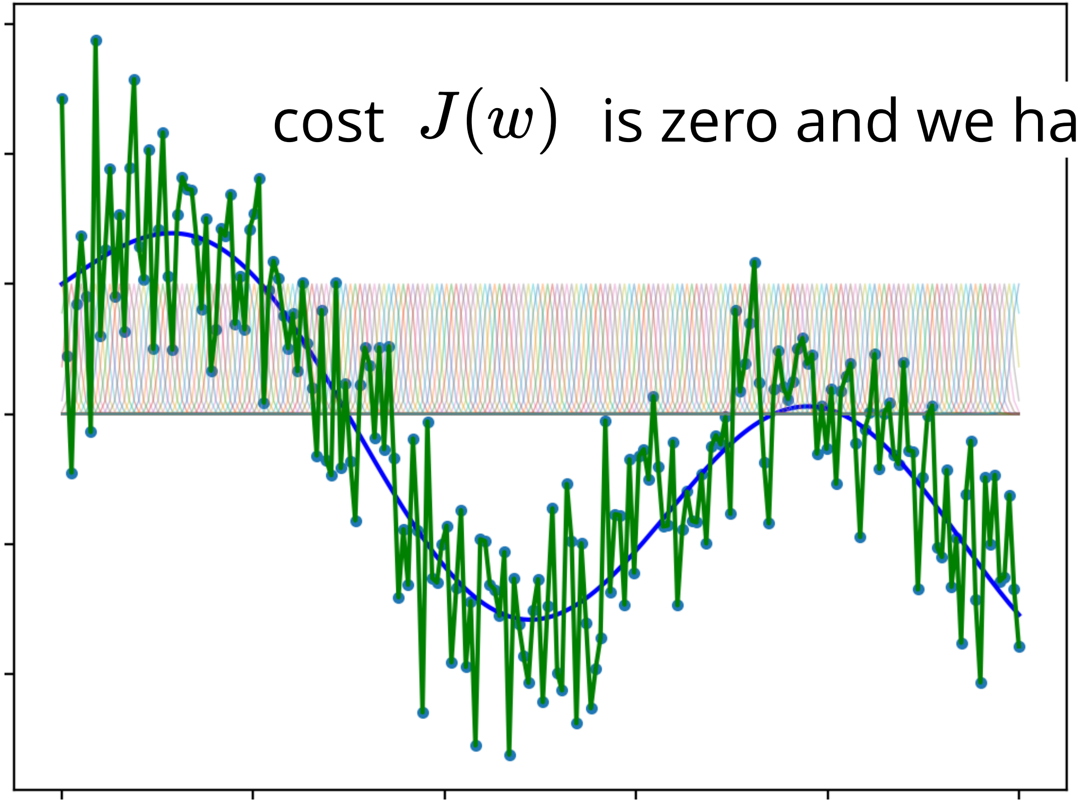
why not more?

using 50 bases!

Example: Gaussian bases



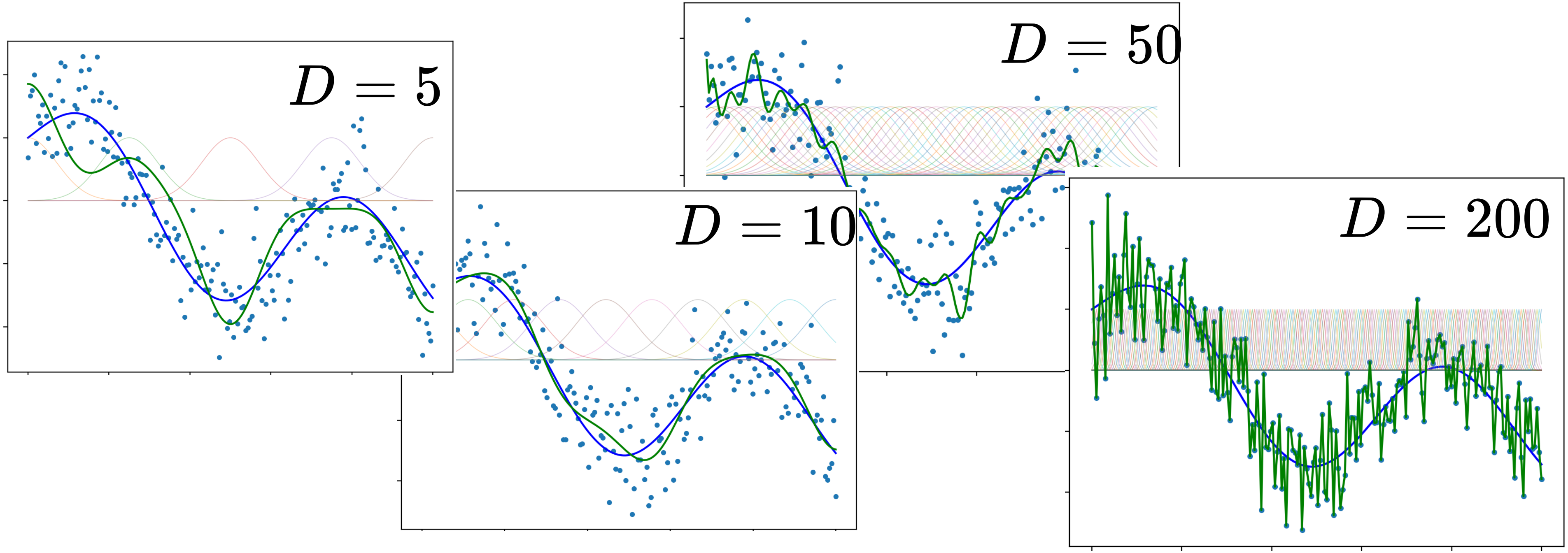
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



cost $J(w)$ is zero and we have a "*perfect*" fit!

using 200, thinner bases (s=.1)

Generalization?

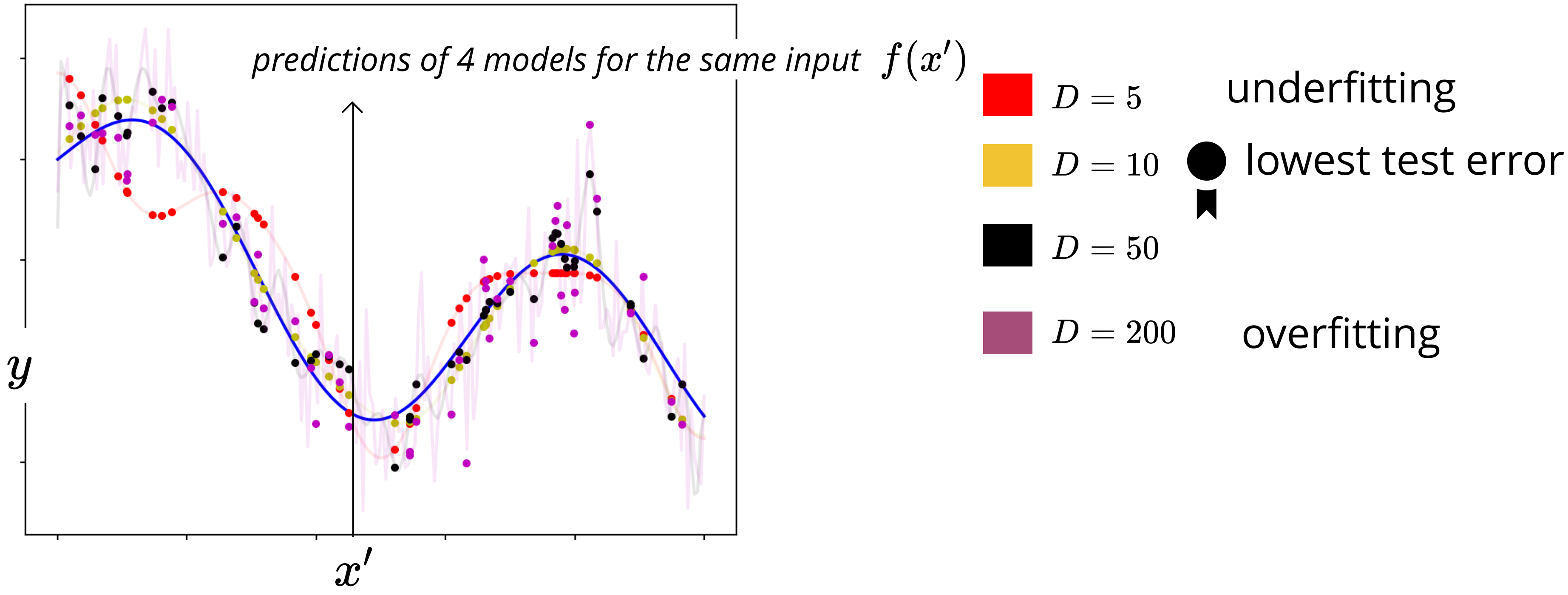


lower training error

which one of these models performs better at test time?

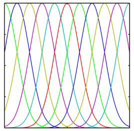
Overfitting

which one of these models performs better at test time?

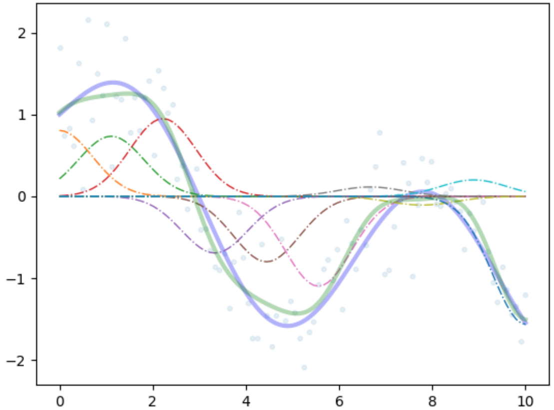


An observation

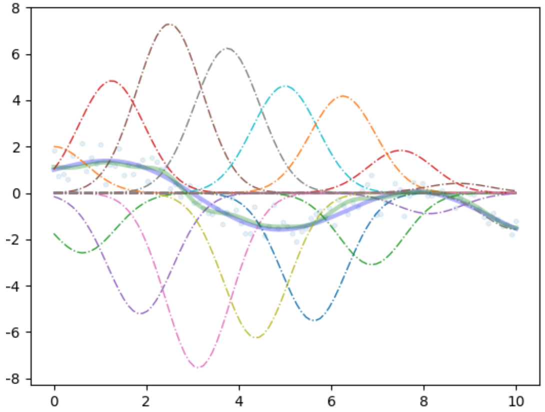
when overfitting, we sometimes see large weights

 dashed lines are $w_d \phi_d(x) \quad \forall d$

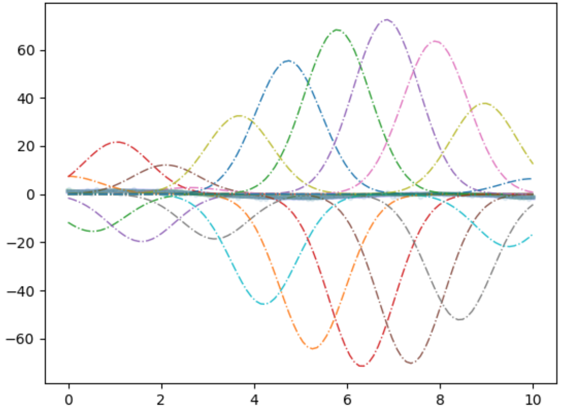
$D = 10$



$D = 17$



$D = 20$



idea: penalize large parameter values

Ridge regression

also known as

L2 regularized linear least squares regression:

$$J(w) = \frac{1}{2}||Xw - y||_2^2 + \frac{\lambda}{2}||w||_2^2$$

sum of squared error

$$\frac{1}{2}\sum_n (y^{(n)} - w^\top x)^2$$

squared L2 norm of w

$$w^T w = \sum_d w_d^2$$

regularization parameter $\lambda > 0$ controls the strength of regularization

a good practice is to **not** penalize the intercept $\lambda(||w||_2^2 - w_0^2)$

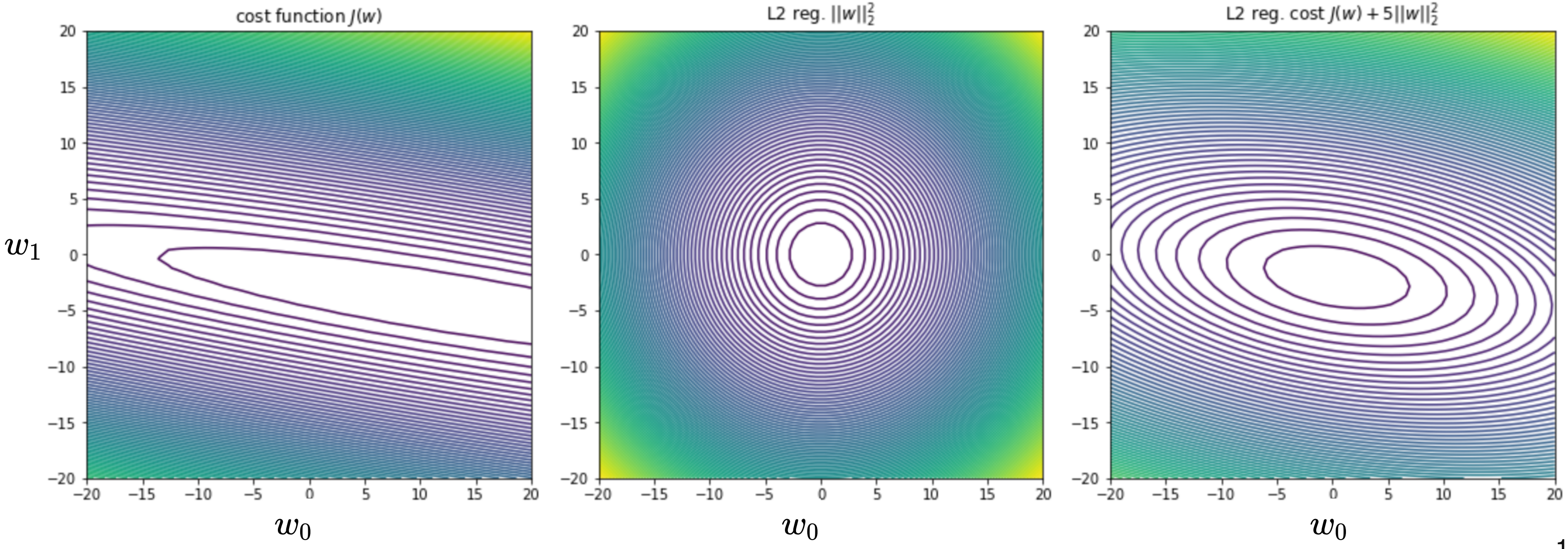
λ is a hyper-parameter (use a validation set or cross-validation to pick the best value)

Ridge regression

example

Visualizing the effect of regularization on the cost function

is the new cost function convex? $\frac{1}{2N}\sum_{x,y\in\mathcal{D}}(y - w^\top x)^2 + \frac{\lambda}{2}||w||_2^2$



Ridge regression

set the derivative to zero

$$J(w) = \frac{1}{2}\sum_{x,y\in\mathcal{D}}(y - w^\top x)^2 + \frac{\lambda}{2}w^\top w$$

$$\nabla J(w) = \sum_{x,y\in\mathcal{D}} x(w^\top x - y) + \lambda w$$

$$= X^\top(Xw - y) + \lambda w = 0$$

when using gradient descent, this term reduces the weights at each step **(weight decay)**

linear system of equations $(X^\top X + \lambda\mathbf{I})w = X^\top y$

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

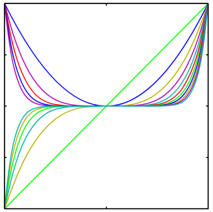
the only part different due to regularization

λI makes it invertible, adds a small value to the diagonals $X^\top X$

we can have linearly dependent features

the solution will be unique!

Example: polynomial bases

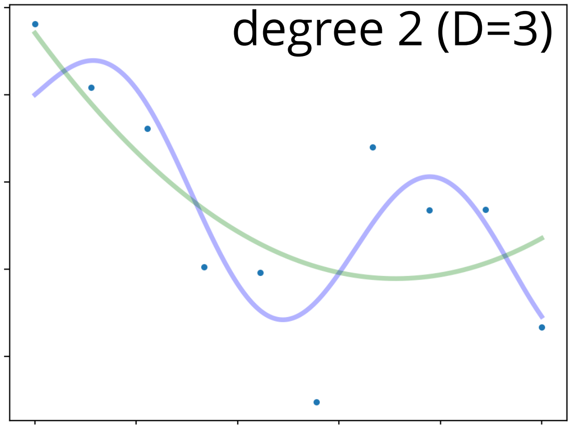


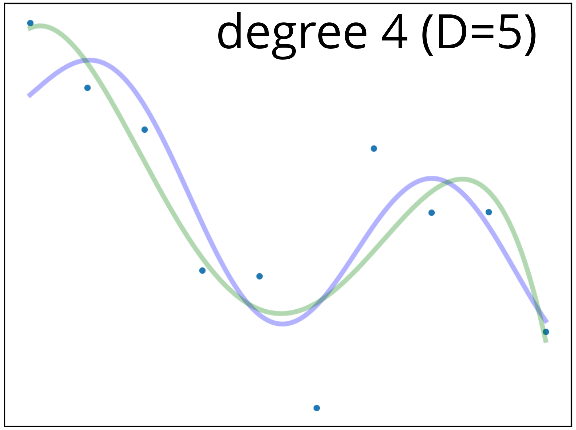
polynomial bases

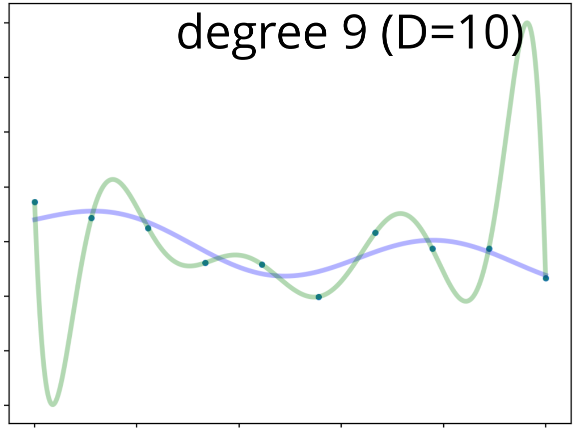
$$\phi_k(x) = x^k$$

Without regularization:

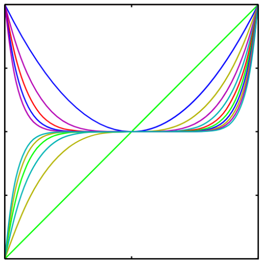
- using D=10 we can perfectly fit the data (high test error)







Example: polynomial bases

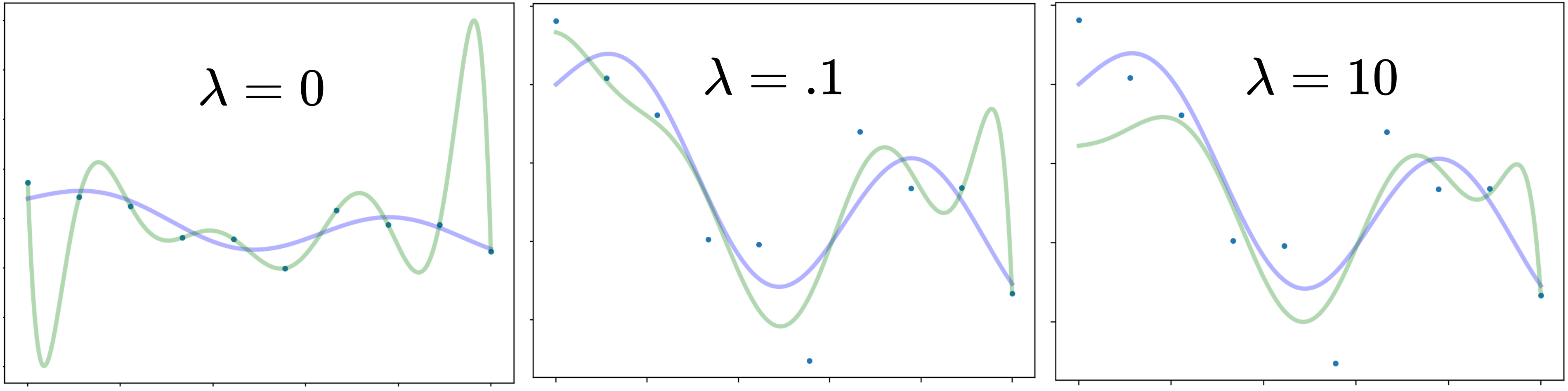


polynomial bases

$$\phi_k(x) = x^k$$

with regularization:

- fixed D=10, changing the amount of regularization



Probabilistic interpretation

recall linear regression & logistic regression maximize log-likelihood

$$w^{MLE} = \arg\max_w p(y|X, w)$$

linear regression $w^{MLE} = \arg\max_w \prod_{x,y\in\mathcal{D}} \mathcal{N}(y|w^\top x, \sigma^2)$

logistic regression $w^{MLE} = \arg\max_w \prod_{x,y\in\mathcal{D}} \text{Bernoulli}(y; \sigma(w^\top x))$

can we do Bayesian inference instead of maximum likelihood?

$$p(w|y, X) \propto p(w)p(y|w, X)$$

posterior prior likelihood

Maximum a Posteriori (MAP)

can we do Bayesian inference instead of maximum likelihood?

$$p(w|y, X) \propto p(w)p(y|w, X)$$

posterior prior likelihood

in general, this is expensive, but there's a cheap compromise:

MAP estimate $w^{MAP} = \arg\max_w p(w)p(y|X, w)$

$$= \arg\max_w \log p(y|X, w) + \log p(w)$$

likelihood: original objective prior

all that is changing is the additional penalty on w

Gaussian Prior

MAP estimate $w^{MAP} = \arg\max_w \log p(y|X,w) + \log p(w)$

prior

assume independent zero-mean Gaussians

$\mathcal{N}(\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

$$\log p(w) = \log \prod_{d=1}^{D} \mathcal{N}(w_d|0,\tau^2) = -\sum_d \frac{w^2}{2\tau^2} + \text{const.}$$

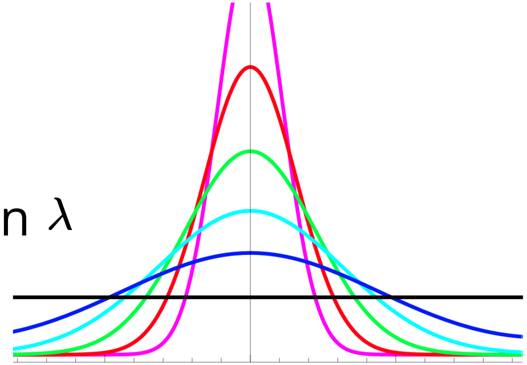
does not depend on w

so it doesn't affect the optimization

lets call $\frac{1}{\tau^2} \to \lambda$

we get the L2 regularization penalty $\frac{\lambda}{2}||w||_2^2$

smaller variance of the prior τ^2 gives larger regularization λ



Laplace prior

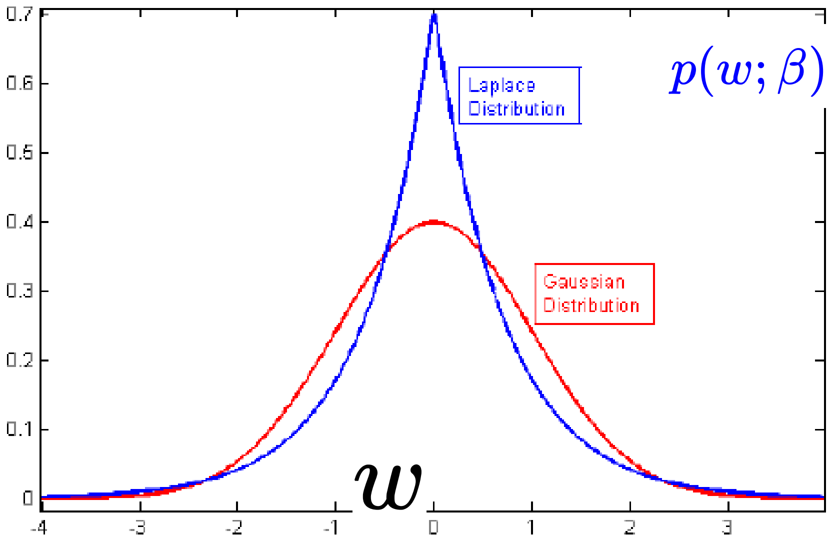
another notable choice of prior is the Laplace distribution

minimizing negative log-likelihood $\rightarrow$ $\sum_d \log p(w_d) = -\sum_d \frac{1}{\beta}|w_d| = -\frac{1}{\beta}||w||_1$

L1 norm of w

L1 regularization: $J(w) \leftarrow J(w) + \lambda||w||_1$ also called lasso

(least absolute shrinkage and selection operator)



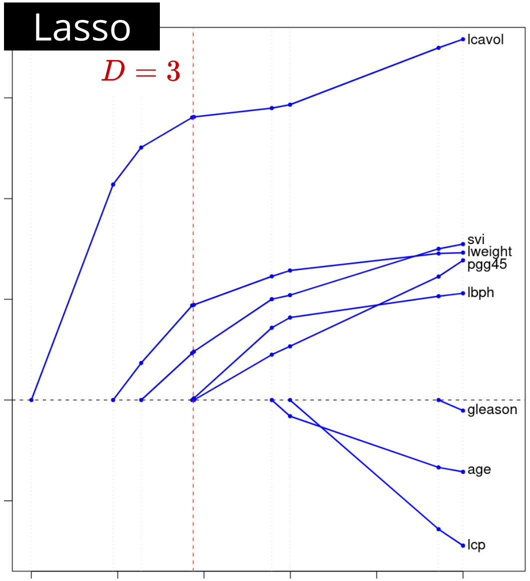
$p(w;\beta) = \frac{1}{2\beta}e^{-\frac{|w|}{\beta}}$ notice the peak around zero

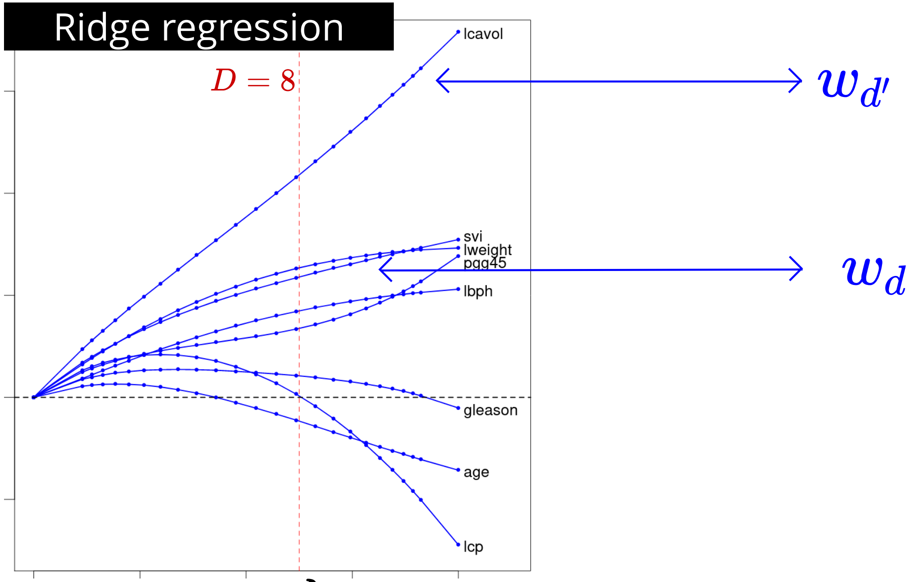
image from here

L_1 vs L_2 regularization

regularization path shows how $\{w_d\}$ change as we change λ
Lasso produces sparse weights (many are zero, rather than small)
red-line is the optimal λ from cross-validation

Example
$D = 8$





decreasing regularization coef. λ

optional

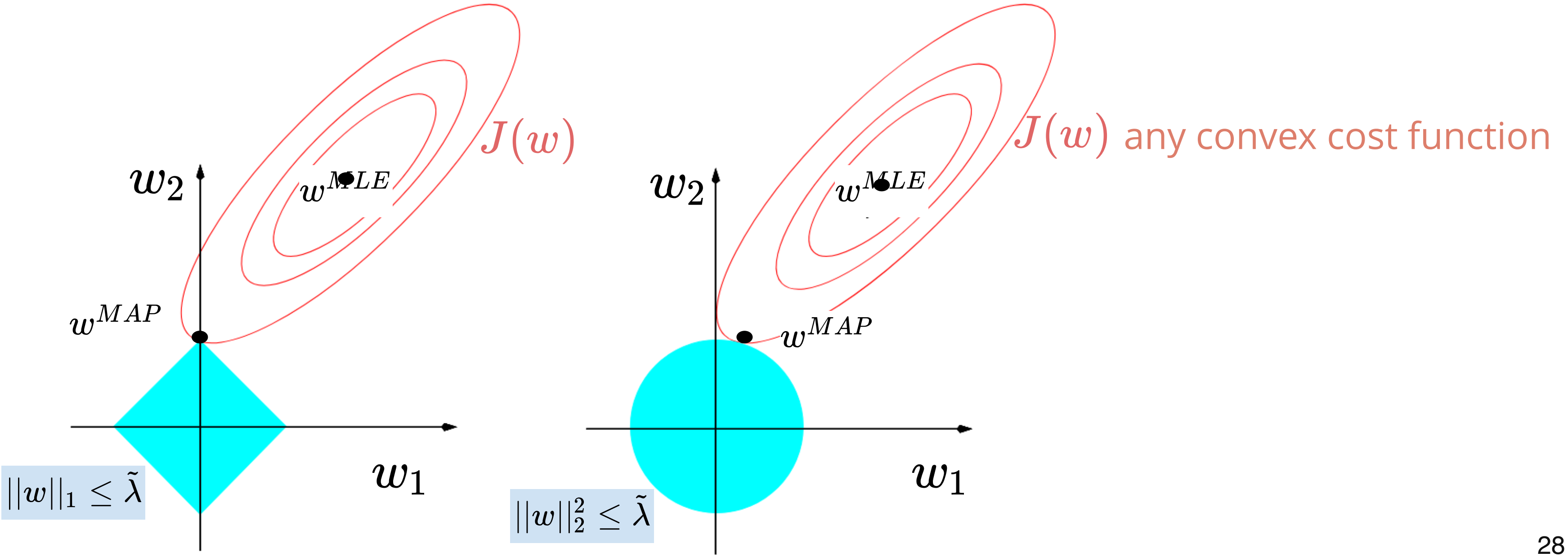
L_1 vs L_2 regularization

$\min_w J(w) + \lambda ||w||_p^p$

is equivalent to $\min_w J(w)$ *subject to* $||w||_p^p \leq \tilde{\lambda}$ for an appropriate choice of $\tilde{\lambda}$

figures below show the constraint and the isocontours of $J(w)$

optimal solution with L1-regularization is more likely to have zero components

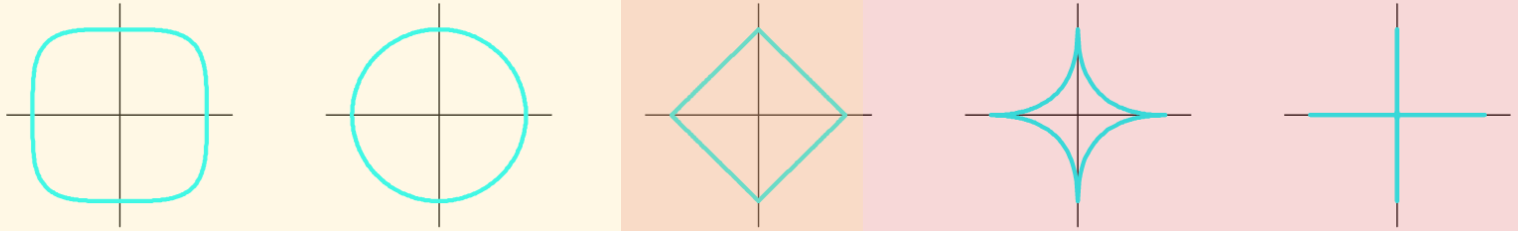


optional

Subset selection

p-norms with $p \geq 1$ are convex (easier to optimize)

p-norms with $p \leq 1$ induces sparsity



$\left(\sum_d w_d^4\right)^{1/4}$ $\left(\sum_d w_d^2\right)^{\frac{1}{2}}$ $\sum_d |w_d|$ $\left(\sum_d |w_d|^{\frac{1}{2}}\right)^2$ $\left(\sum_d |w_d|^{\frac{1}{10}}\right)^{10}$

L_0 norm — closer to **0-norm** →

penalizes the **number of** features with non-zero weights

$$J(w) + \lambda ||w||_0 = J(w) + \lambda \sum_d \mathbb{I}(w_d \neq 0)$$

a penalty of λ for each feature to be included in the model

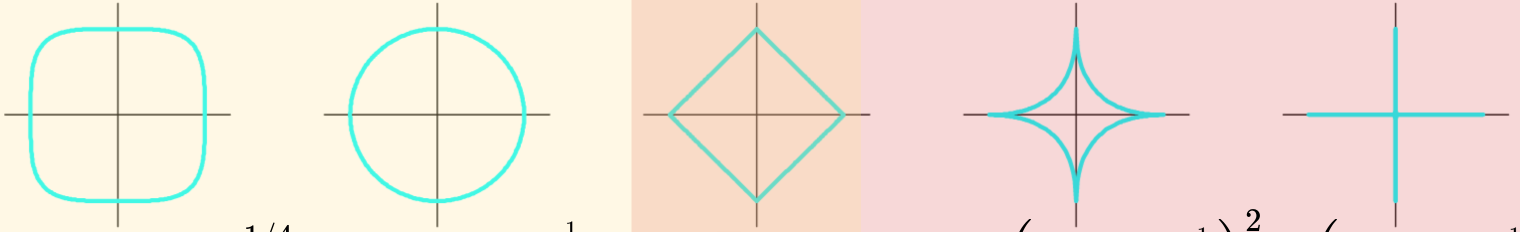
performs feature selection

optional

Subset selection

p-norms with $p \geq 1$ are convex (easier to optimize)

p-norms with $p \leq 1$ induces sparsity



$\left(\sum_d w_d^4\right)^{1/4}$ $\left(\sum_d w_d^2\right)^{\frac{1}{2}}$ $\sum_d |w_d|$ $\left(\sum_d |w_d|^{\frac{1}{2}}\right)^2$ $\left(\sum_d |w_d|^{\frac{1}{10}}\right)^{10}$

L_0 norm ——— closer to **0-norm** →

optimizing this is a difficult *combinatorial problem*:

- search over all 2^D subsets

L1 regularization is a viable alternative to L0 regularization

Adding L_2 regularization

do not penalize the bias w_0

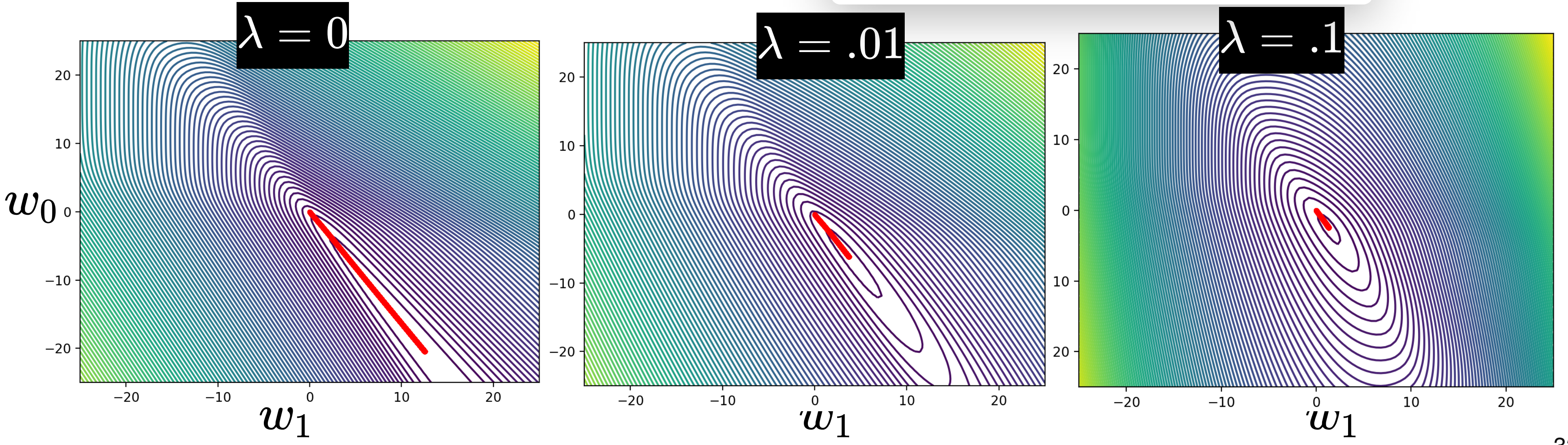
L2 penalty makes the optimization easier too!

note that the optimal w_1 shrinks

example for **logistic regression**

```
def gradient(x, y, w, lambdaa):
    N,D = x.shape
    yh = logistic(np.dot(x, w))
    grad = np.dot(x.T, yh - y) / N
    grad[1:] += lambdaa * w[1:]
    return grad
```

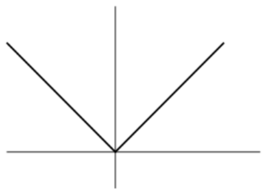
weight decay



similar pattern for **linear regression**, see example in the colab

optional

Subgderivatives



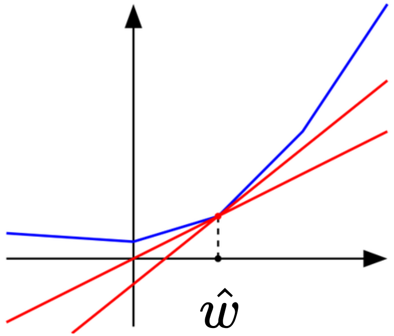
L1 penalty is no longer smooth or differentiable (at 0)

extend the notion of derivative to non-smooth functions

sub-differential is the set of all **sub-derivatives** at a point

$$\partial f(\hat{w}) = \left[\lim_{w \to \hat{w}^-} \frac{f(w)-f(\hat{w})}{w-\hat{w}}, \lim_{w \to \hat{w}^+} \frac{f(w)-f(\hat{w})}{w-\hat{w}}\right]$$

if $\boldsymbol{f}$ is differentiable at $\hat{w}$ then sub-differential has one member $\frac{d}{dw} f(\hat{w})$



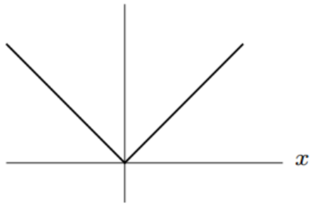
another expression for sub-differential

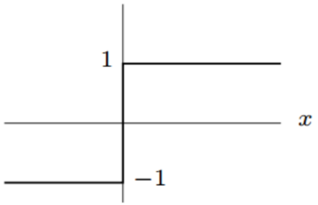
$$\partial f(\hat{w}) = \{g \in \mathbb{R} | \; f(w) > f(\hat{w}) + g(w - \hat{w})\}$$

optional

Subgradient

example subdifferential for $f(w) = |w|$





$$\partial f(0) = [-1, 1]$$

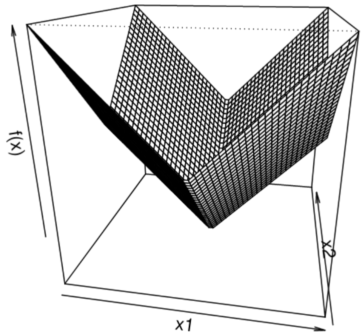
$$\partial f(w \neq 0) = \{\text{sign}(w)\}$$

recall, **gradient** was the vector of **partial derivatives**

subgradient is a vector of **sub-derivatives**

subdifferential for functions of multiple variables

$$\partial f(\hat{w}) = \{g \in \mathbb{R}^D | f(w) > f(\hat{w}) + g^\top (w - \hat{w})\}$$



we can use sub-gradient with diminishing step-size for optimization

image credit: G. Gordon

Adding L_1 regularization

L1-regularized *linear regression* has efficient solvers

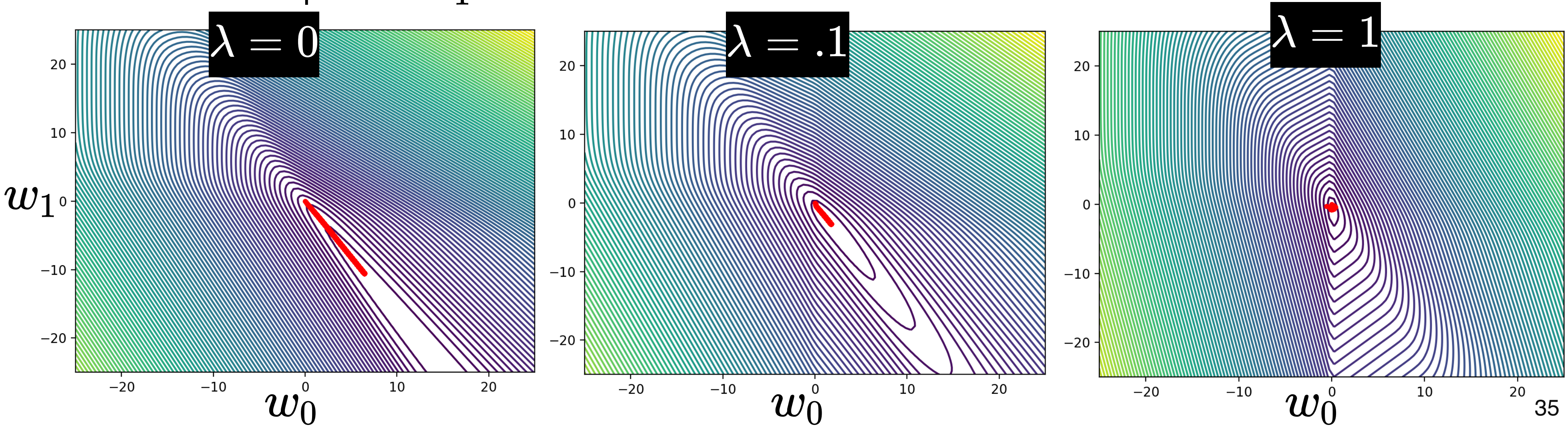
subgradient method for L1-regularized logistic regression

do not penalize the bias w_0

using **diminishing learning rate**

note that the optimal w_1 **becomes 0**

```python
def gradient(x, y, w, lambdaa):
    N,D = x.shape
    yh = logistic(np.dot(x, w))
    grad = np.dot(x.T, yh - y) / N
    grad[1:] += lambdaa * np.sign(w[1:])
    return grad
```



Regularization serves many purposes

$$w^* = (X^T X)^{-1} X^T y$$

$D \times 1$ $\quad D \times N \quad N \times D \quad N \times 1$

what if $X^T X$ is **not invertible**?

add a small value to the diagonals, a.k.a. **regularize**

what if **linear fit is not the best**?

use nonlinear basis

How to avoid **overfitting** then? **regularize**

what if **we want a sparse model**?

do feature selection and only keep important parameters with **regularizing**

optional

Data normalization

what if we scale the input features, using different factors $\tilde{x}_d^{(n)} = \gamma_d x_d^{(n)} \forall d, n$

if we have no regularization: $\tilde{w}_d = \frac{1}{\gamma_d} w_d \forall d$

everything remains the same because: $||Xw - y||_2^2 = ||\tilde{X}\tilde{w} - y||_2^2$

with regularization: $||\tilde{w}||_2 \neq ||w||_2^2$ so the optimal **w** will be different!

features of different mean and variance will be penalized differently

normalization

$$\begin{cases} \mu_d = \frac{1}{N} x_d^{(n)} \\ \sigma_d^2 = \frac{1}{N-1}(x_d^{(n)} - \mu_d)^2 \end{cases}$$

makes sure all features have the same mean and variance $x_d^{(n)} \leftarrow \frac{x_d^{(n)} - \mu_d}{\sigma_d}$

we saw that this also helps with the optimization!

Generalization and model complexity

simple models cannot fit the data

- large training error due to underfitting

bias

expressive models can overfit the data

- small training error
- large test error due to overfitting

variance

regularization can help us trade-off between bias and variance

we want to see how these two terms contribute to the **generalization error**

Generalization and model complexity

example

columns: a different type of model $g(x)$

rows: different datasets

datasets are from the same distribution

$$x^{(n)}, y^{(n)} \sim p(x, y)$$

$F(x)$ the best possible model

higher variance

the complex model varies more with the dataset
it may not generalize well for this reason

higher bias

the simple model is biased to a particular type of data
it underfits, but it has a low variance

$g(x) = \text{fixed}$ $\quad g(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$ $\quad g(x) = w_0 + w_1 x$

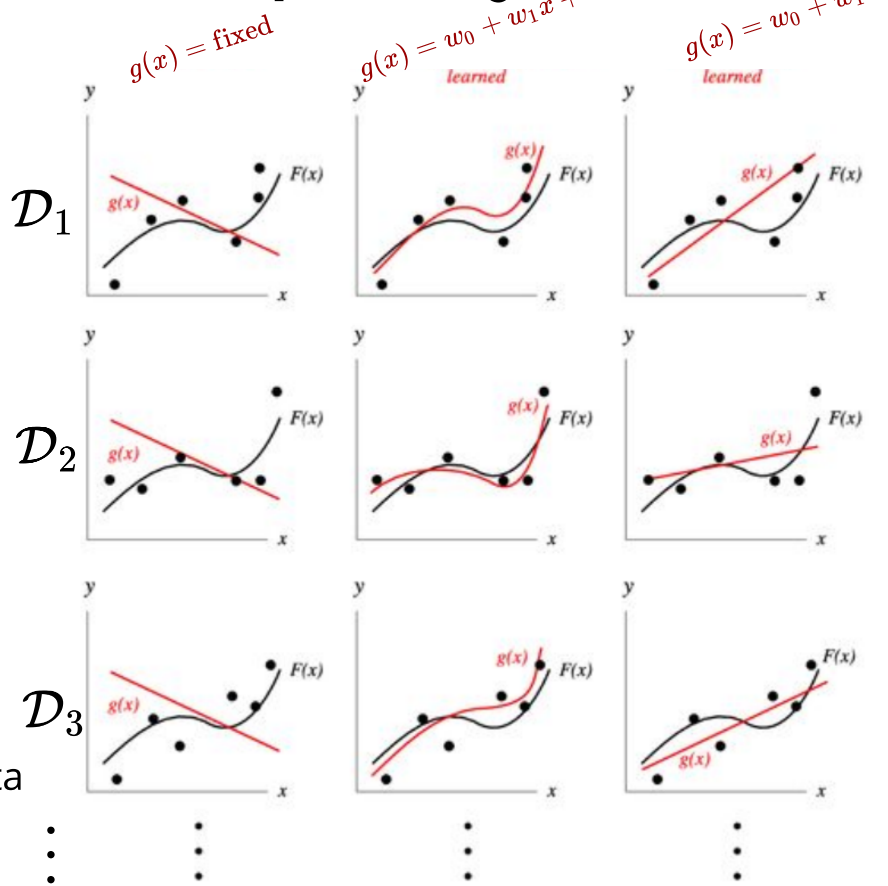


image:Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*.

Bias-variance decomposition: Setup

decompose the generalization error to see the effect of bias and variance (for L2 loss)

assume a true distribution $p(x, y)$

best prediction given L2 loss $f(x) = \mathbb{E}_p[y|x]$ (saw this in k-means and regression trees as well!)

assume that a dataset $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_n$ is sampled from $p(x, y)$

let $\hat{f}_\mathcal{D}$ be our model based on the dataset

what we care about is the generalization error (aka expected loss, expected risk)

$$\mathbb{E}[(\hat{f}_\mathcal{D}(x) - y)^2]$$

all blue items are random variables

Bias-variance decomposition

what we care about is the generalization error

$$\mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - y)^2] \quad = \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - y + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]$$

$f(x) + \epsilon$

$\hat{f}_{\mathcal{D}}(x) + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]$ add and subtract a term

above simplifies to the following (the remaining terms are going to be zero)

$$= \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] \quad + \mathbb{E}[\epsilon^2]$$

variance

bias^2

unavoidable noise error

Bias-variance decomposition

the expected loss is decomposed to:

$$= \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[\epsilon^2]$$

variance: how change of dataset affects the prediction

bias: how average over all datasets differs from the regression function

noise error: the error even if we used the true model $f(x)$

different models vary in their trade off between error due to bias and variance

- simple models: often more biased
- complex models: often have more variance

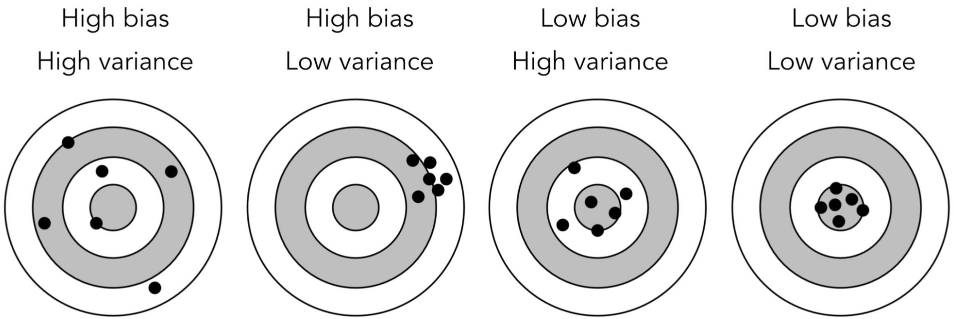


image from here

Bias vs. variance

example

distribution of error (cost) due to randomness of dataset

we care about the expected error

bias causes a high error for all choices of dataset

higher variance also increases the expected error

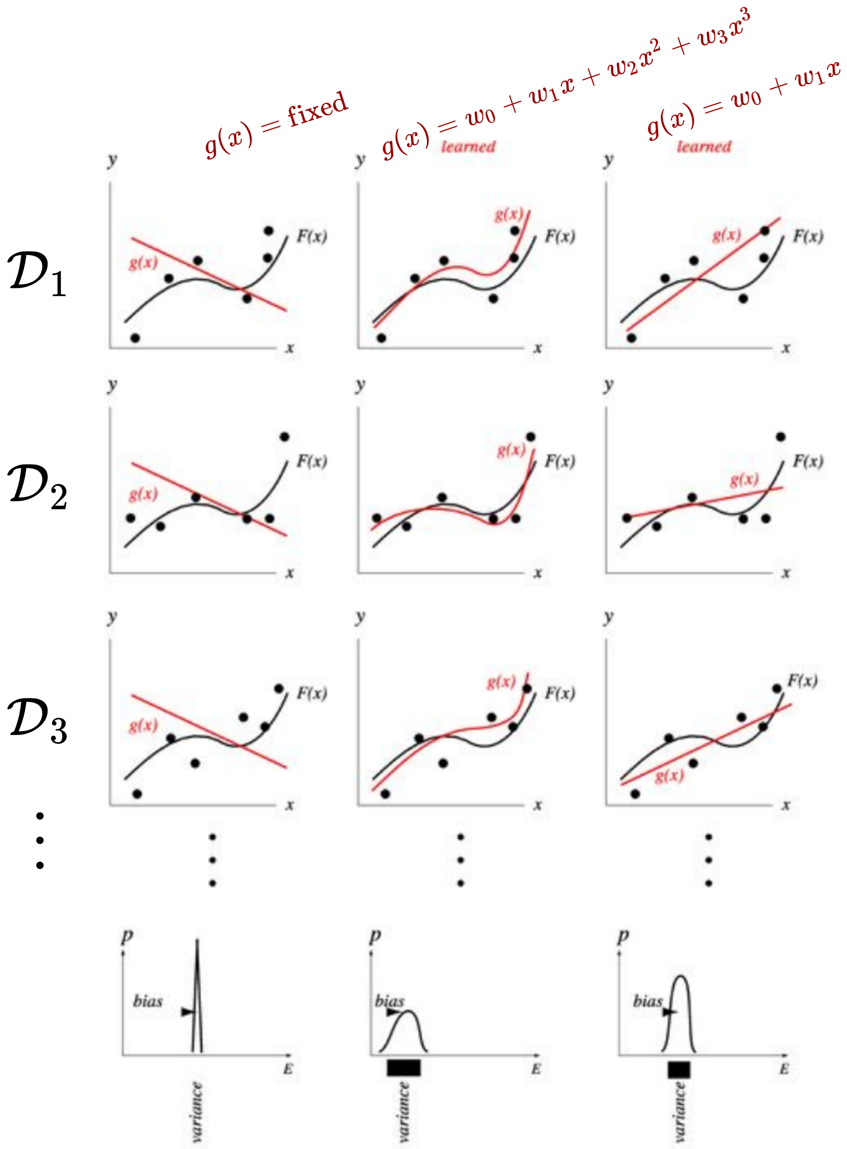


image:Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*.

Example: bias vs. variance

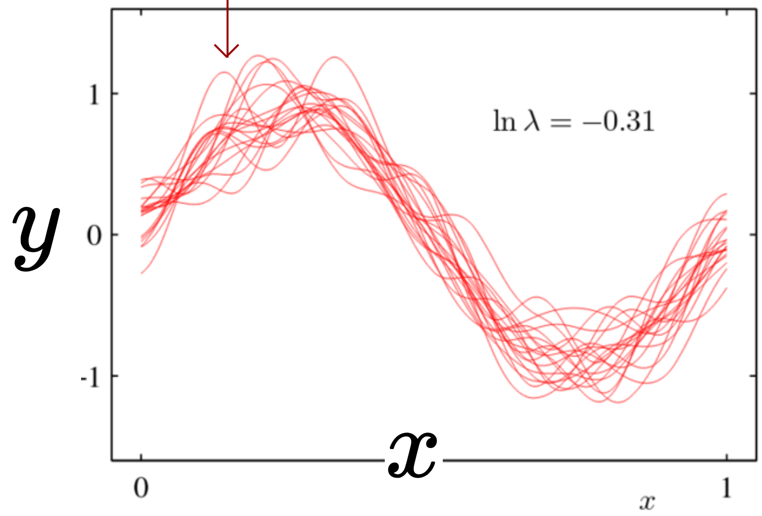
models for different datasets $\hat{f}_{\mathcal{D}}$

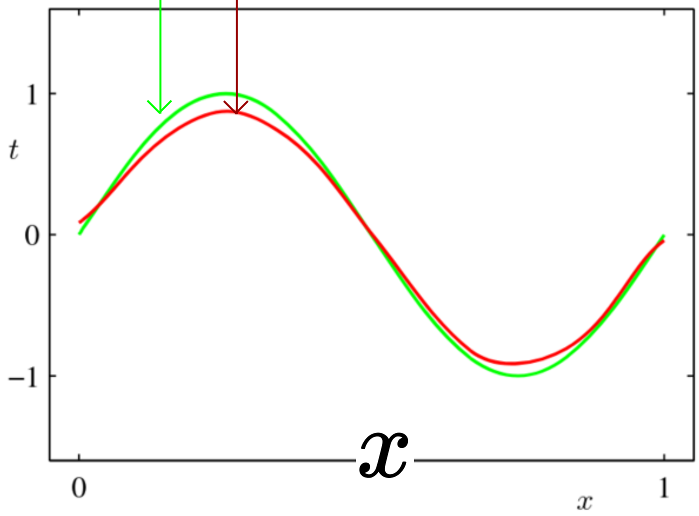
using Gaussian bases

random datasets of size N=25 instances are not shown

true model f

their average $\mathbb{E}[\hat{f}_{\mathcal{D}}]$

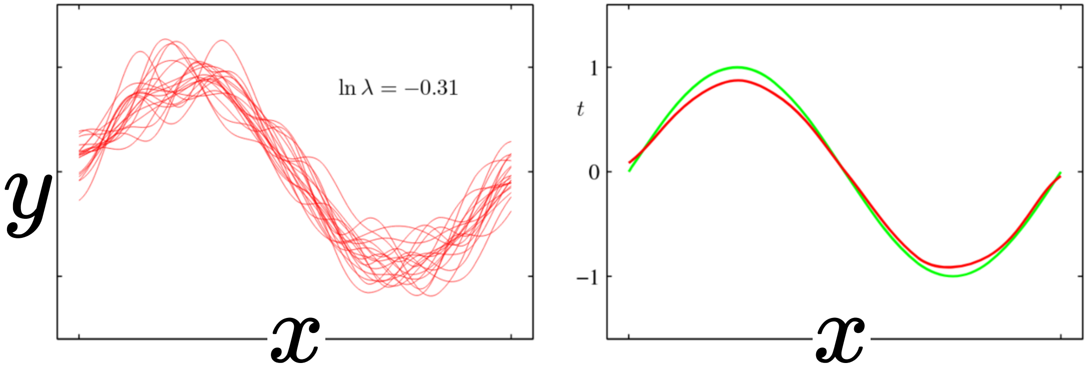




variance is the average difference (in squared L2 norm) between these curves and their average

bias is the difference (in L2 norm) between two curves

Example: bias vs. variance

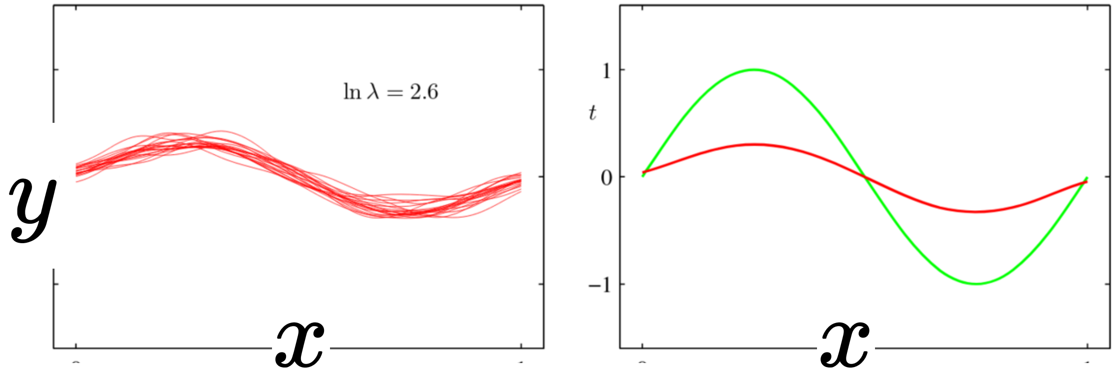


side note

the average fit is very good, despite high variance

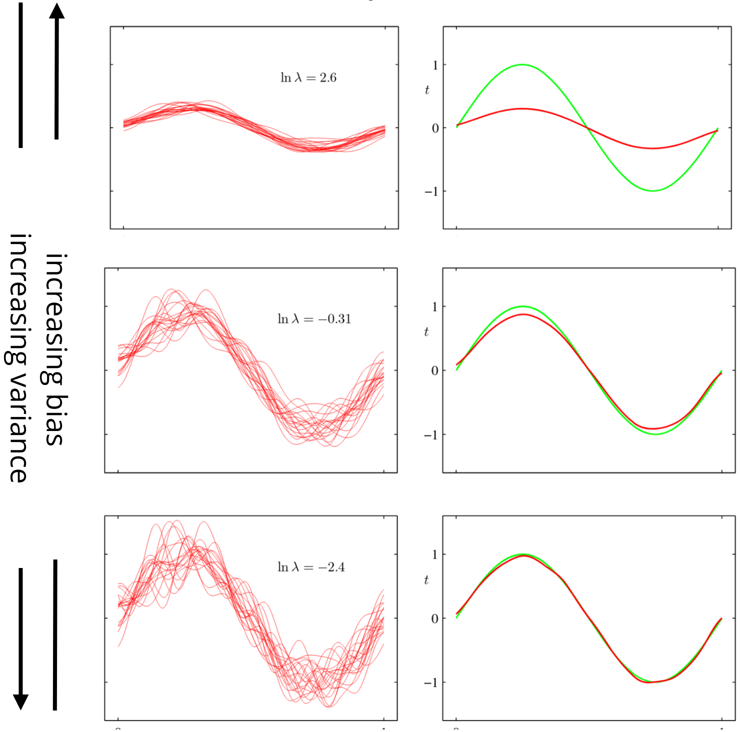
model averaging: uses "average" prediction of expressive models to prevent overfitting

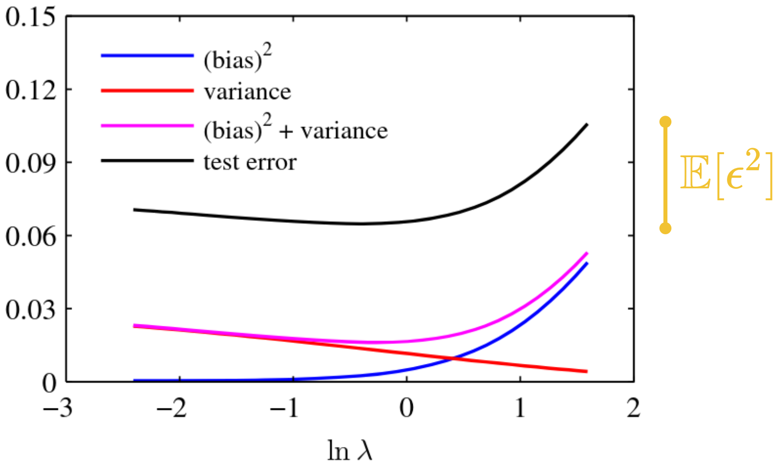
using larger regularization penalty: higher bias - lower variance



Example: bias vs. variance

the lowest expected loss (test error) is somewhere between the two extremes

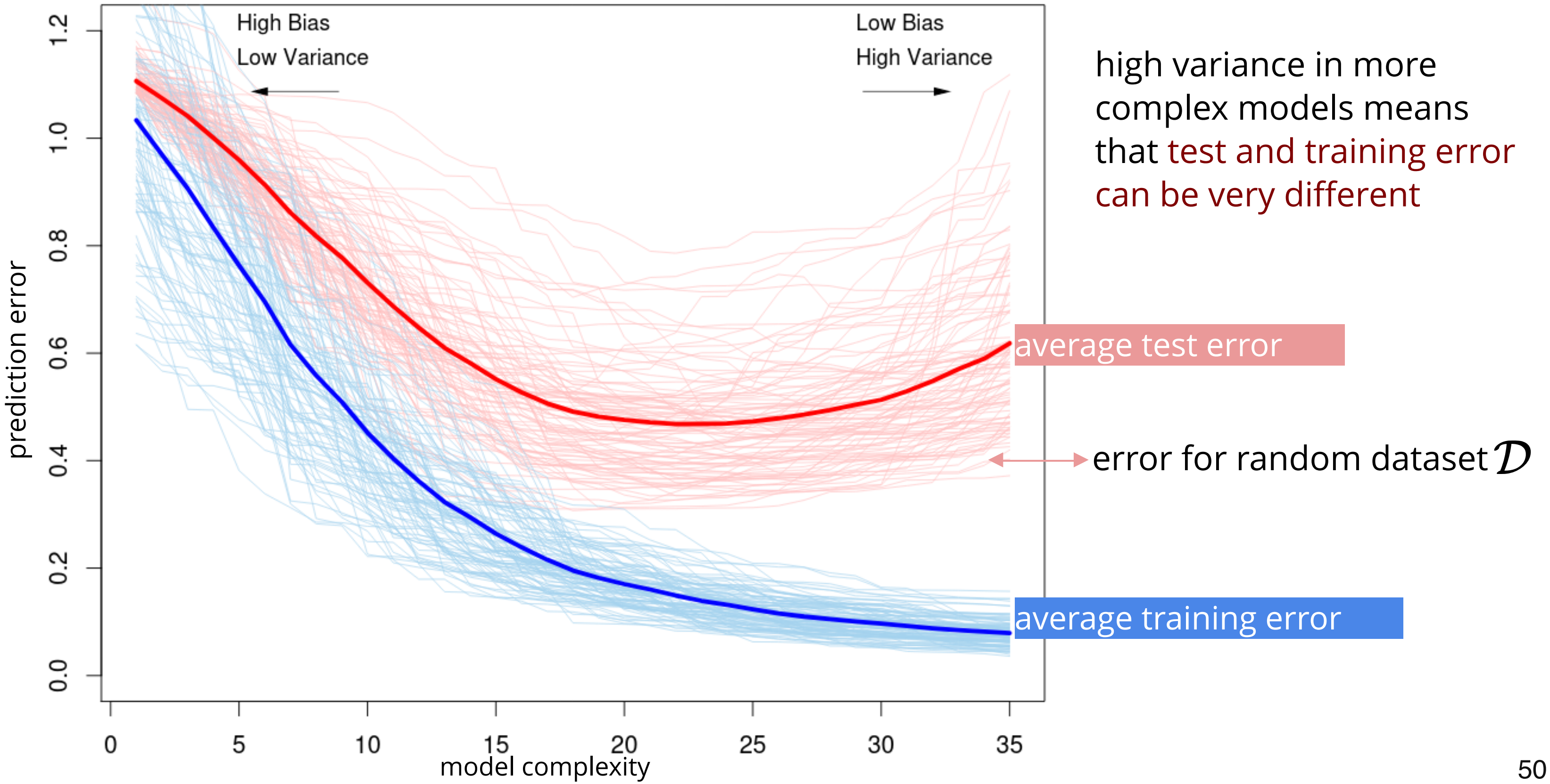




in practice, how to decide which model to use?

Effect on training and test error

high bias in simplistic models means that training error can be high



high variance in more complex models means that test and training error can be very different

Summary

- complex models can have very different training and test error (*generalization gap*)
- regularization bounds this gap by penalizing model complexity
  - L1 & L2 regularization
  - probabilistic interpretation: different priors on weights
  - L1 produces sparse solutions (useful for feature selection)
- bias-variance trade off:
  - formalizes the relation between
    - training error (bias)
    - complexity (variance) and
    - and the test error (bias + variance)
  - not so elegant beyond L2 loss