Applied Machine Learning

Regularization

Reihaneh Rabbany

School of Computer Science

Learning objectives

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

recall

Linear regression

model: $\hat{y} = f_w(x) = w^\top x \ : \mathbb{R}^D \rightarrow \mathbb{R}$

cost function: $J_w = \frac{1}{N}\sum_n \frac{1}{2}(y^{(n)} - \hat{y}^{(n)})^2 = \frac{1}{2}||y - Xw||^2$

how to find w^*? closed form solution: $w^* = (X^\top X)^{-1} X^\top y$

Or use gradient descent

partial derivatives: $\frac{\partial}{\partial w_d} J_w = \frac{1}{N}\sum_n (\hat{y}^{(n)} - y^{(n)}) x_d^{(n)}$

gradient (all partial derivatives): $\nabla J(w) = \frac{1}{N}\sum_n (\hat{y}^{(n)} - y^{(n)}) x^{(n)} = \frac{1}{N} X^\top (\hat{y} - y)$

optimization with gradient descent: repeat until stopping criterion: $w^{\{t+1\}} \leftarrow w^{\{t\}} - \alpha \nabla J(w^{\{t\}})$

what if **linear fit is not the best**?
how to increase the model's expressiveness?
$\Rightarrow$ use nonlinear basis to 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

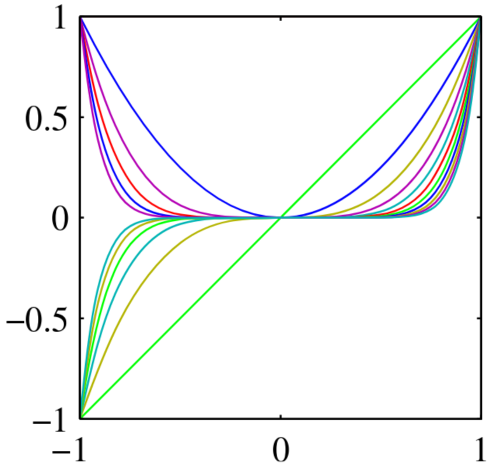
$$
\Phi = \begin{bmatrix}
\phi_1(x^{(1)}), & \phi_2(x^{(1)}), & \cdots, & \phi_D(x^{(1)}) \\
\phi_1(x^{(2)}), & \phi_2(x^{(2)}), & \cdots, & \phi_D(x^{(2)}) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_1(x^{(N)}), & \phi_2(x^{(N)}), & \cdots, & \phi_D(x^{(N)})
\end{bmatrix}
$$

one instance

recall

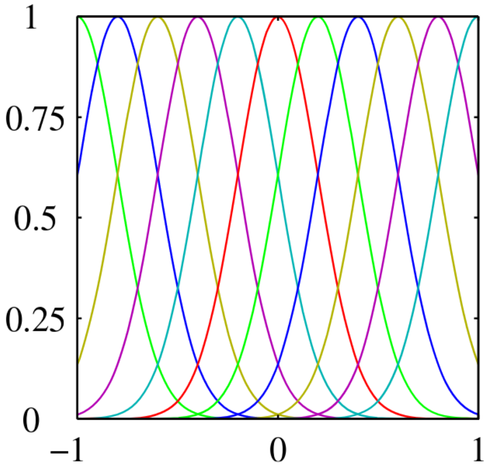
Nonlinear basis functions

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



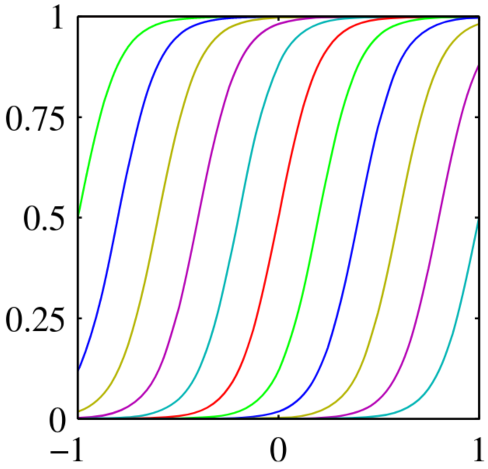
polynomial bases

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



Gaussian bases

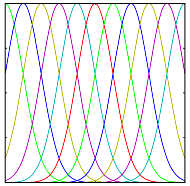
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



Sigmoid bases

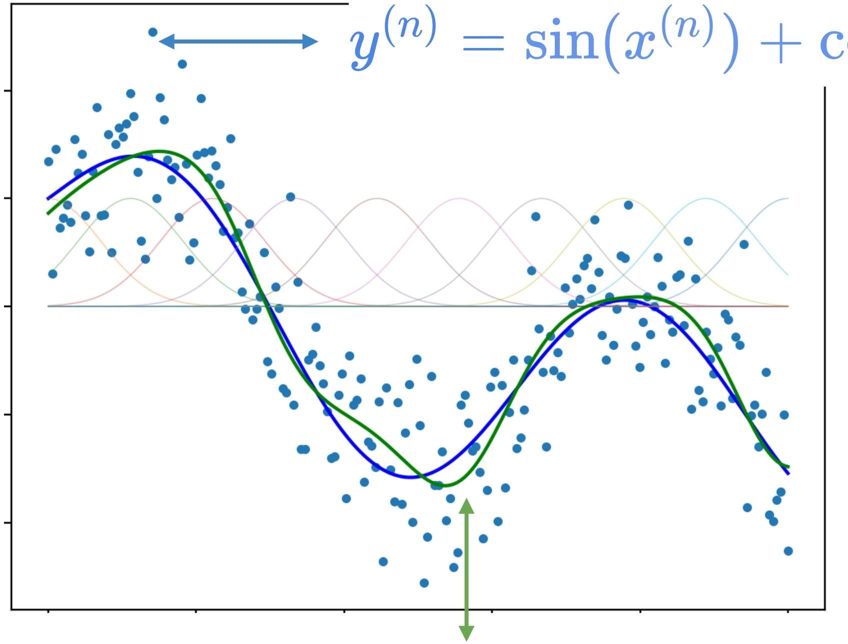
$$\phi_k(x) = \frac{1}{1+e^{-\frac{x-\mu_k}{s}}}$$

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

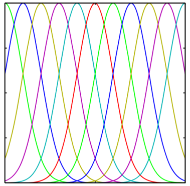
$$f(x') = \phi(x')^\top \underbrace{(\Phi^\top \Phi)^{-1}\Phi^\top y}_{w}$$

new instance

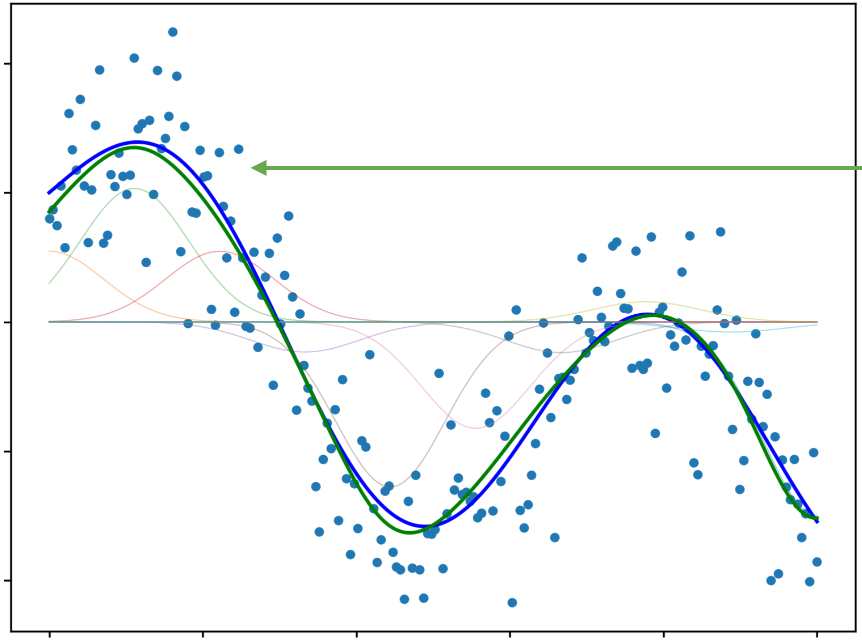
features evaluated for the new point

w found using LLS

Example: Gaussian bases



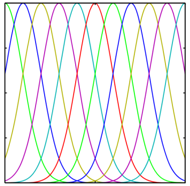
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



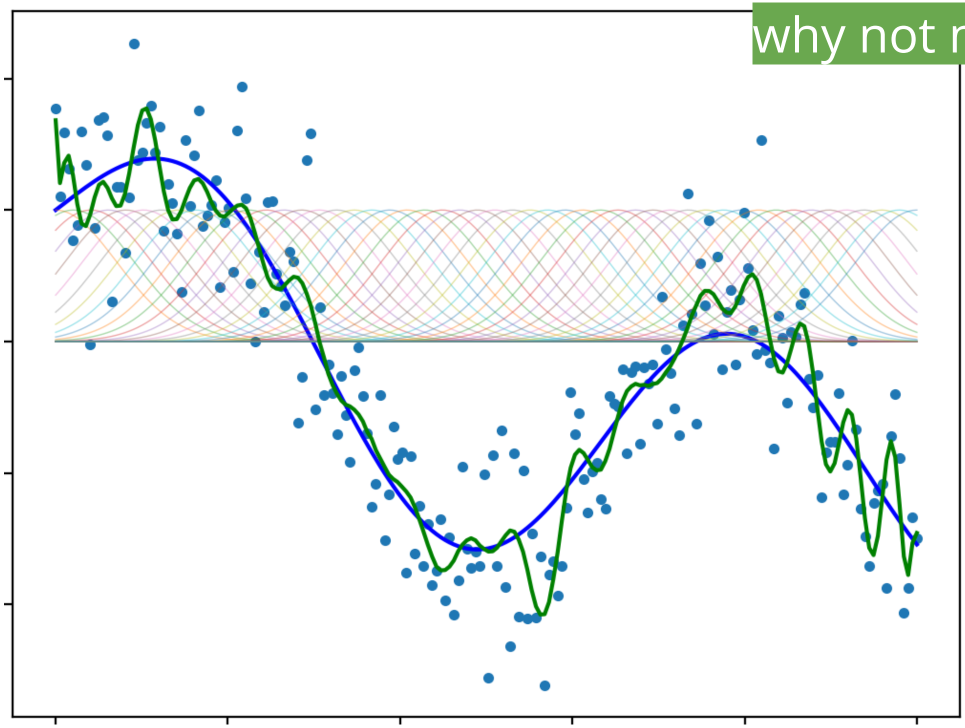
our fit to data using **10 Gaussian bases**

why not more?

Example: Gaussian bases



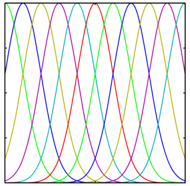
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



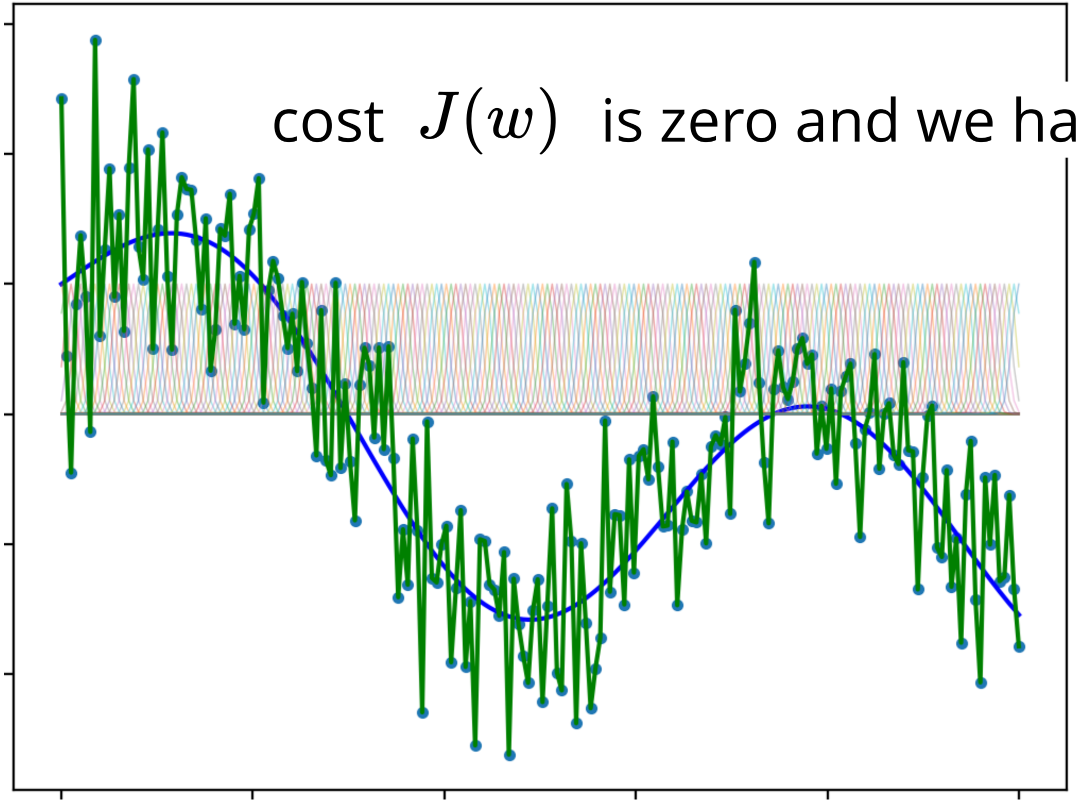
why not more?

using 50 bases!

Example: Gaussian bases



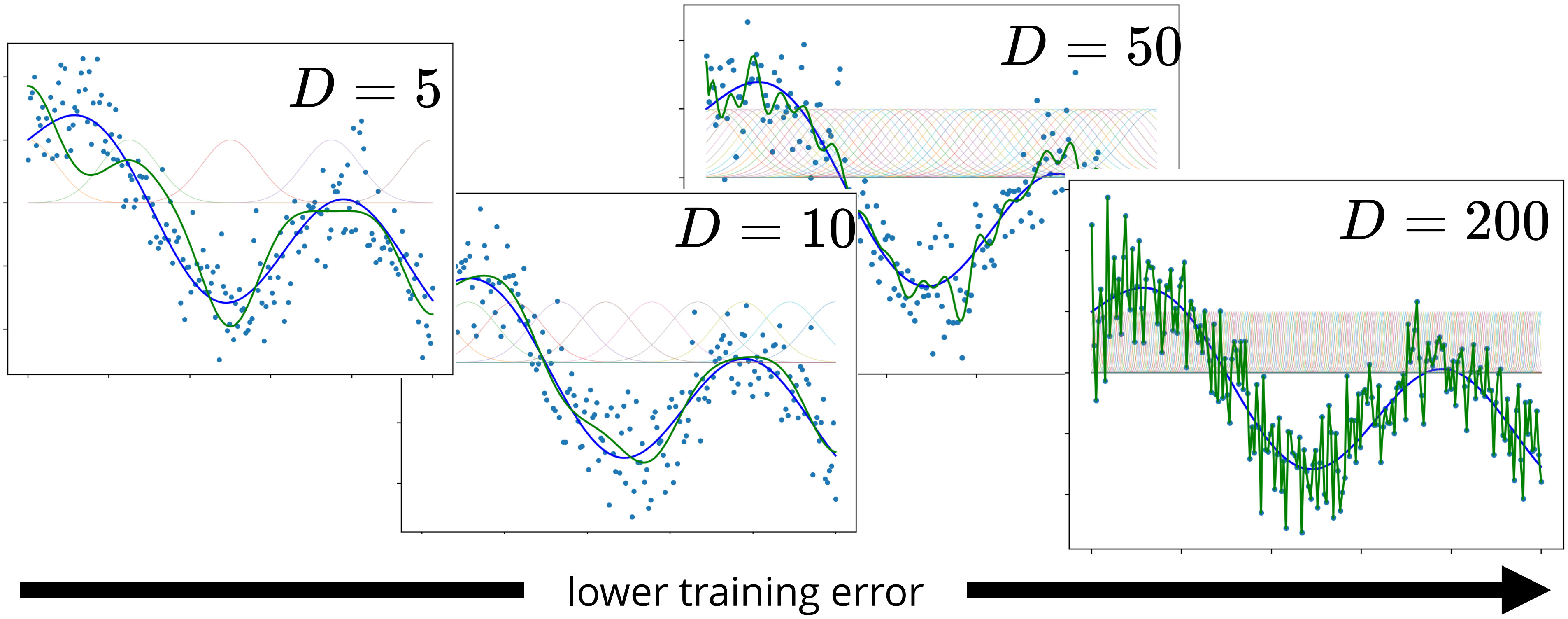
$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



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

using 200, thinner bases (s=.1)

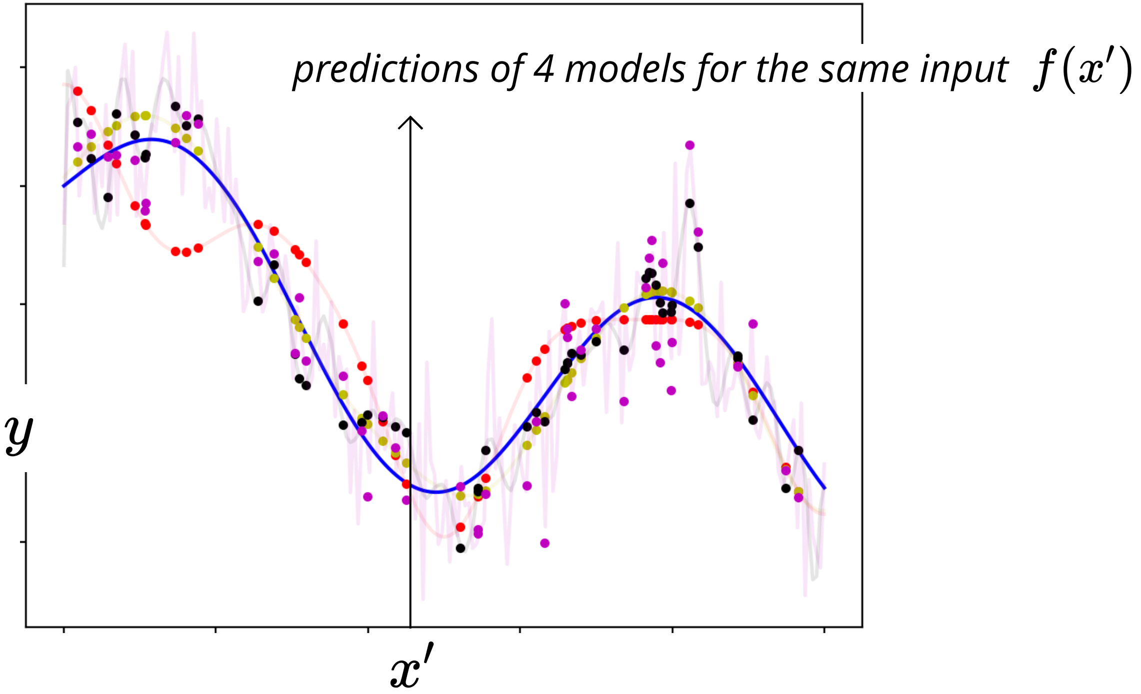
Generalization?



which one of these models performs better at test time?

Overfitting

which one of these models performs better at test time?



$D = 5$ underfitting

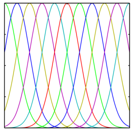
$D = 10$ lowest test error

$D = 50$

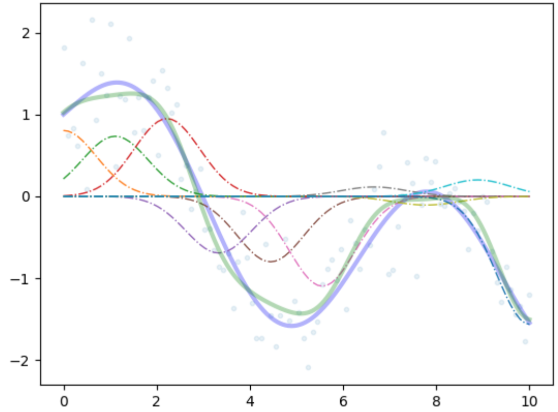
$D = 200$ overfitting

An observation

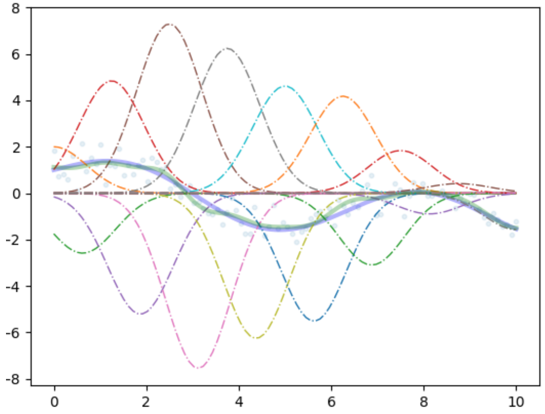
when overfitting, we sometimes see large weights

 dashed lines are $w_d\phi_d(x) \quad \forall d$ $\qquad$ $f_w(x) = \sum_d w_d\, \phi_d(x)$

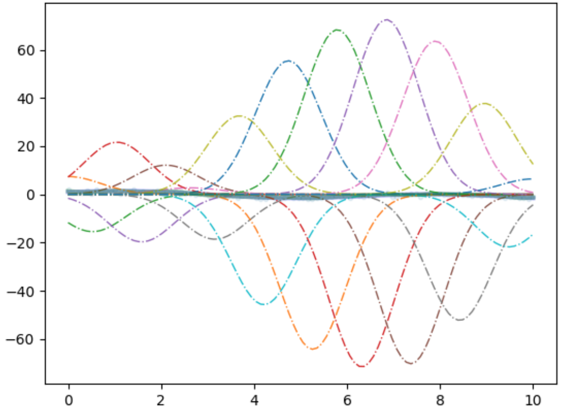
$D = 10$



$D = 17$



$D = 20$



idea: penalize large parameter values

Ridge regression

also known as

L2 regularized linear least squares regression:

$$J(w) = \underbrace{\frac{1}{2}||Xw - y||_2^2}_{\text{sum of squared error}} + \frac{\lambda}{2}\underbrace{||w||_2^2}_{\text{squared L2 norm of w}}$$

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

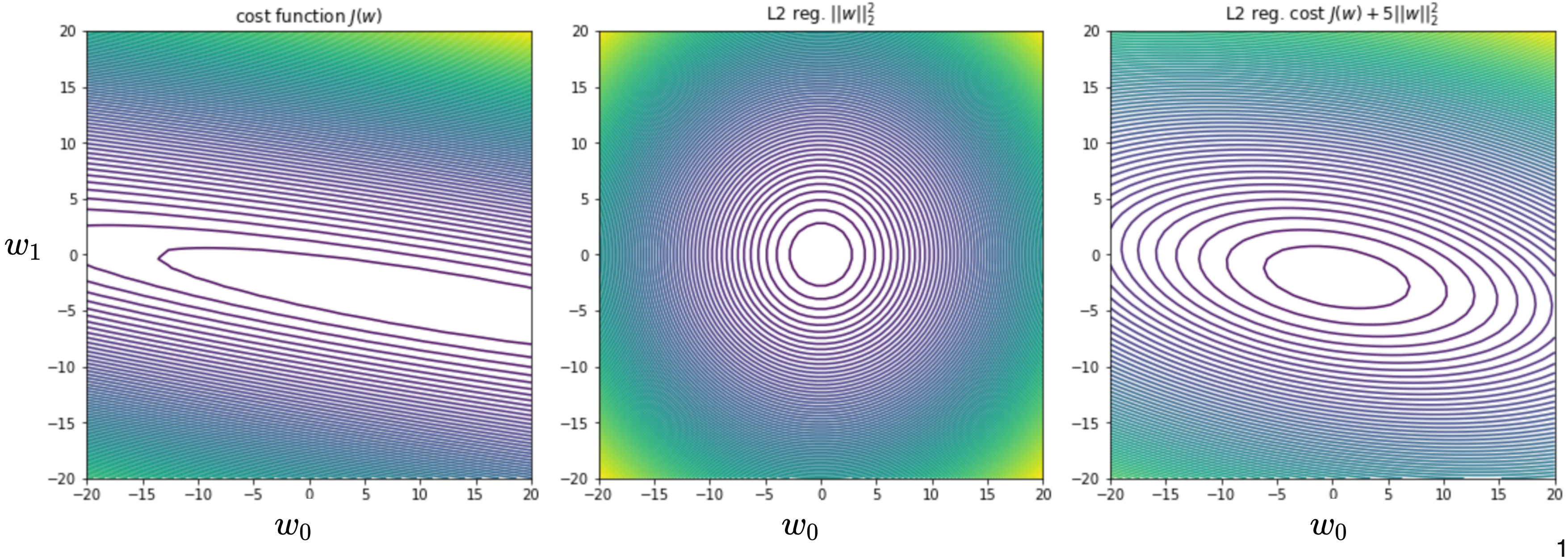
a good practice is to **not** penalize the intercept $\lambda(||w||_2^2 - w_0^2)$

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

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

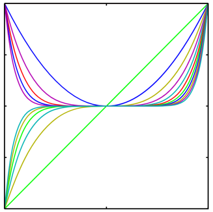
the only part different due to regularization

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

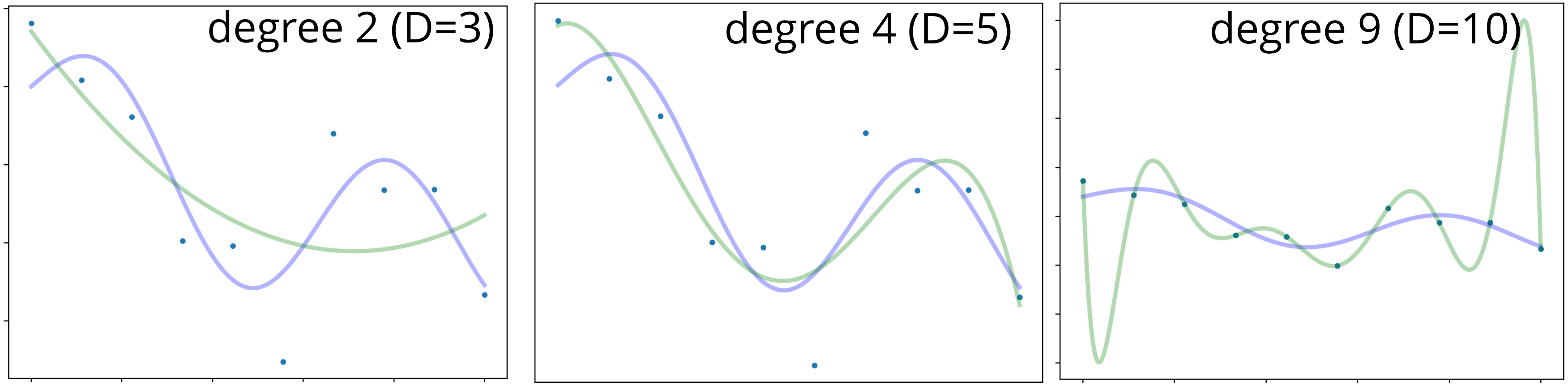


polynomial bases

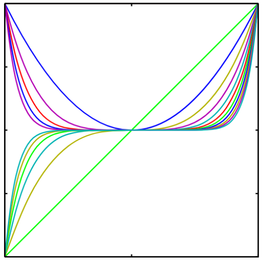
$$\phi_k(x) = x^k$$

Without regularization:

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



Example: polynomial bases

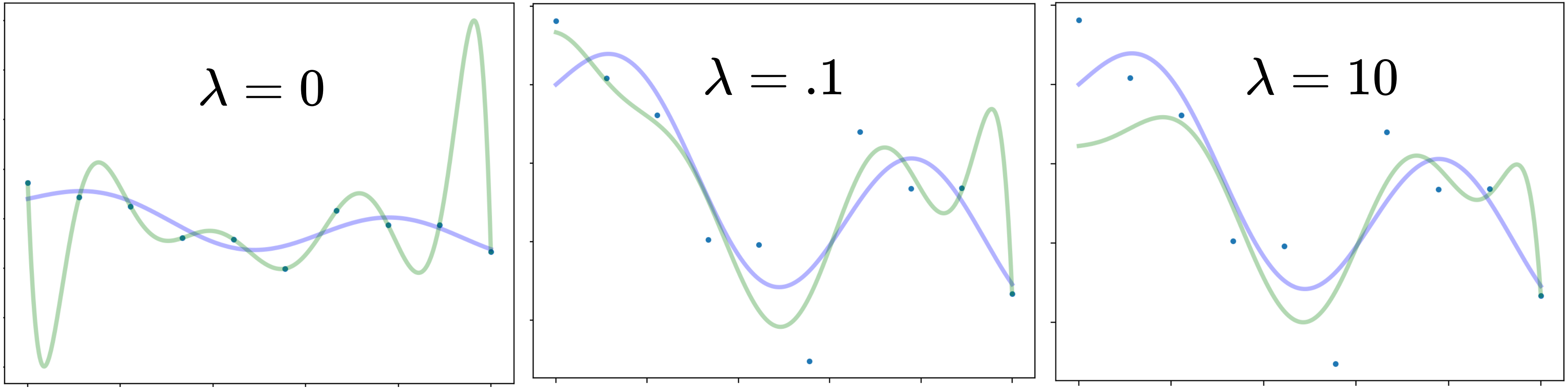


polynomial bases

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

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

logistic regression $\quad w^{MLE} = \arg\max_w \prod_{x,y\in\mathcal{D}} \mathrm{Bernoulli}\left(y|\sigma(w^\top x)\right)$

can we do Bayesian inference instead of maximum likelihood?

$$\underbrace{p(w|y,X)}_{\text{posterior}} \propto \underbrace{p(w)}_{\text{prior}}\underbrace{p(y|w,X)}_{\text{likelihood}}$$

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

prior

assume independent zero-mean Gaussians

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

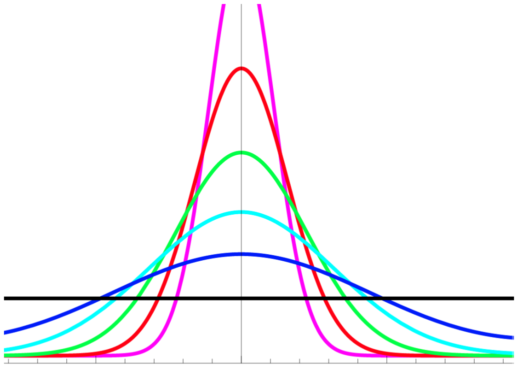
$$\log p(w) = \log \prod_{d=1}^{D} \mathcal{N}(w_d|0, \tau^2) = -\sum_d \frac{w^2}{2\tau^2} + \boxed{\text{const}}.$$

does not depend on w
so it doesn't affect the optimization

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

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

smaller variance of the prior τ gives larger regularization λ



Laplace prior

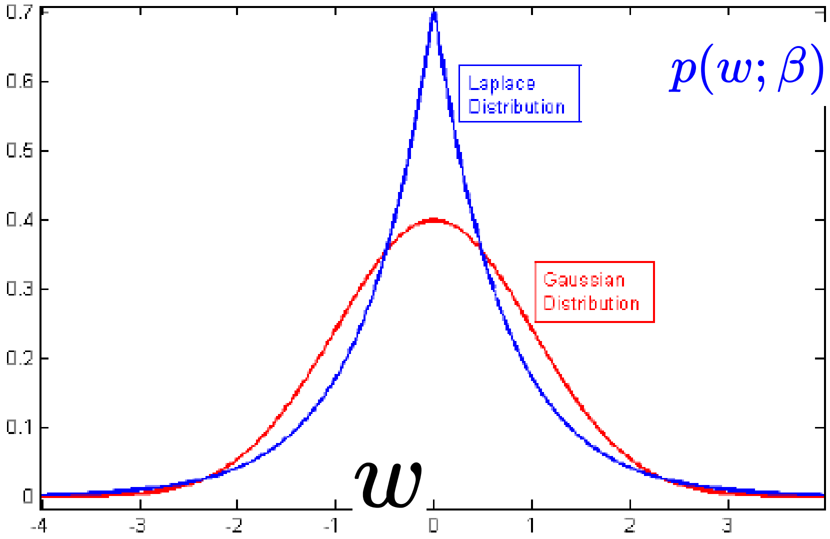
another notable choice of prior is the Laplace distribution

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

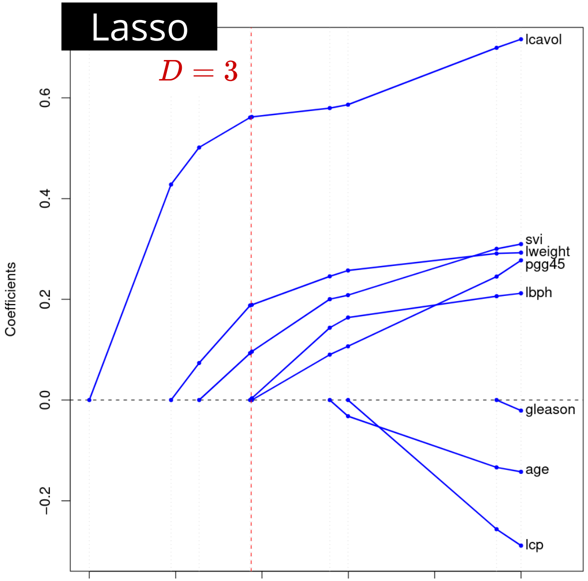
L_1 vs L_2 regularization

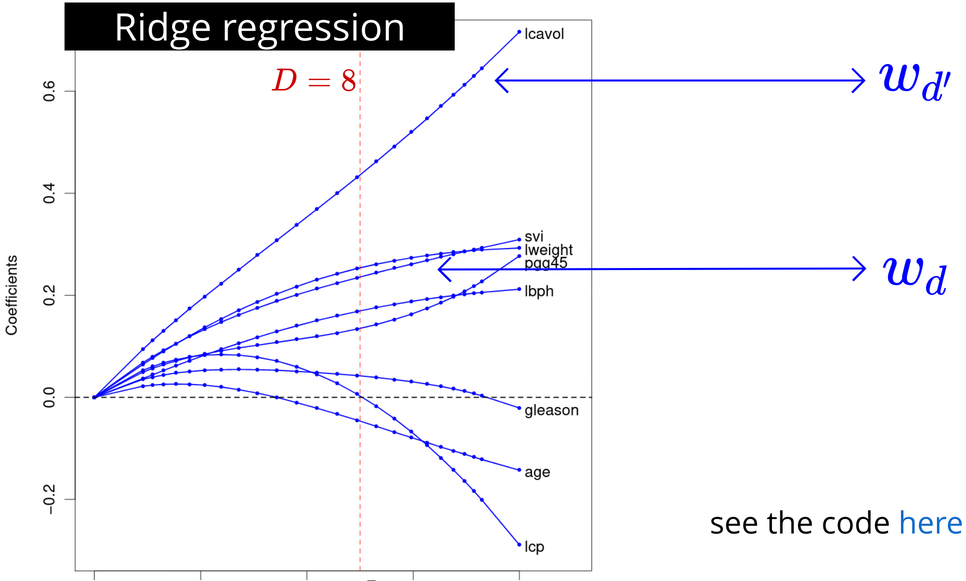
regularization path shows how $\{w_d\}$ change as we change λ

Lasso produces sparse weights (many are zero, rather than small)

Example

$D = 8$





decreasing regularization coef. λ

see the code here

red-line is the optimal λ from cross-validation, for lasso the model uses only 3 of the 8 features

$\Rightarrow$ lasso results in sparse models

optional

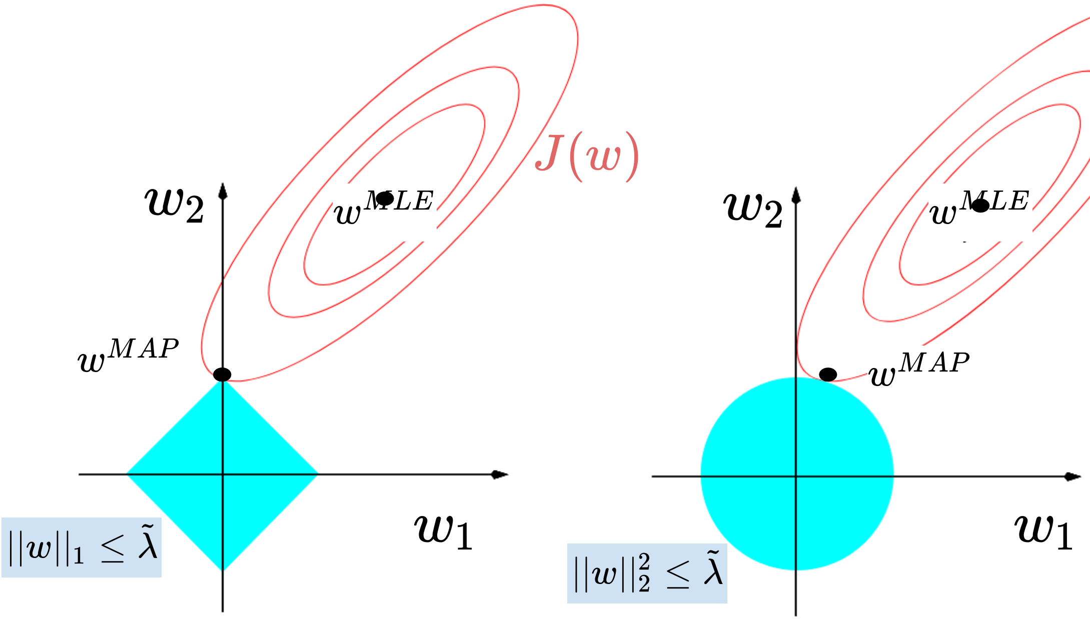
L_1 vs L_2 regularization

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

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



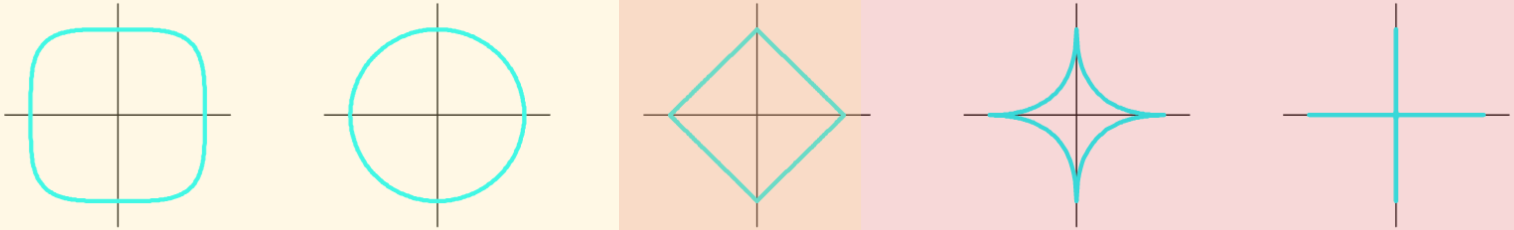
$J(w)$ any convex cost function

optional

Subset selection

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

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



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

closer to **0-norm** → L_0 norm

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

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

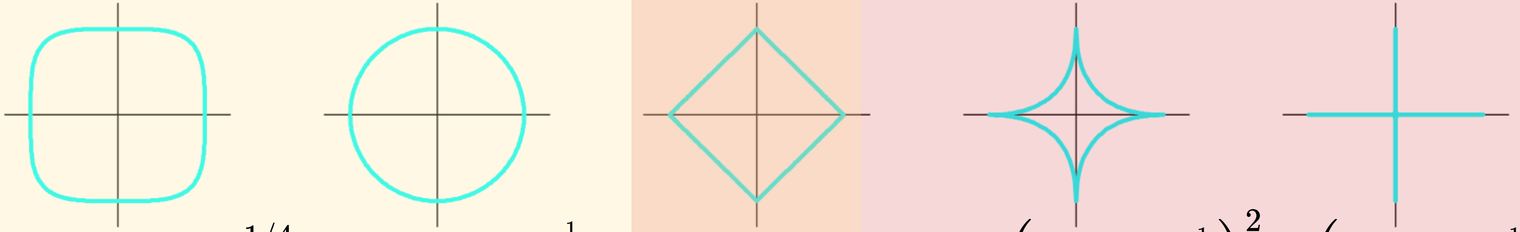
enforces a penalty of λ for each feature to be included in the model $\Rightarrow$ performs feature selection

optional

Subset selection

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

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



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

closer to **0-norm** → L_0 norm

L1 regularization is a viable alternative to L0 regularization

optimizing l_0 regularization is a difficult *combinatorial problem*: search over all 2^D subsets

Adding L_2 regularization

do not penalize the bias w_0

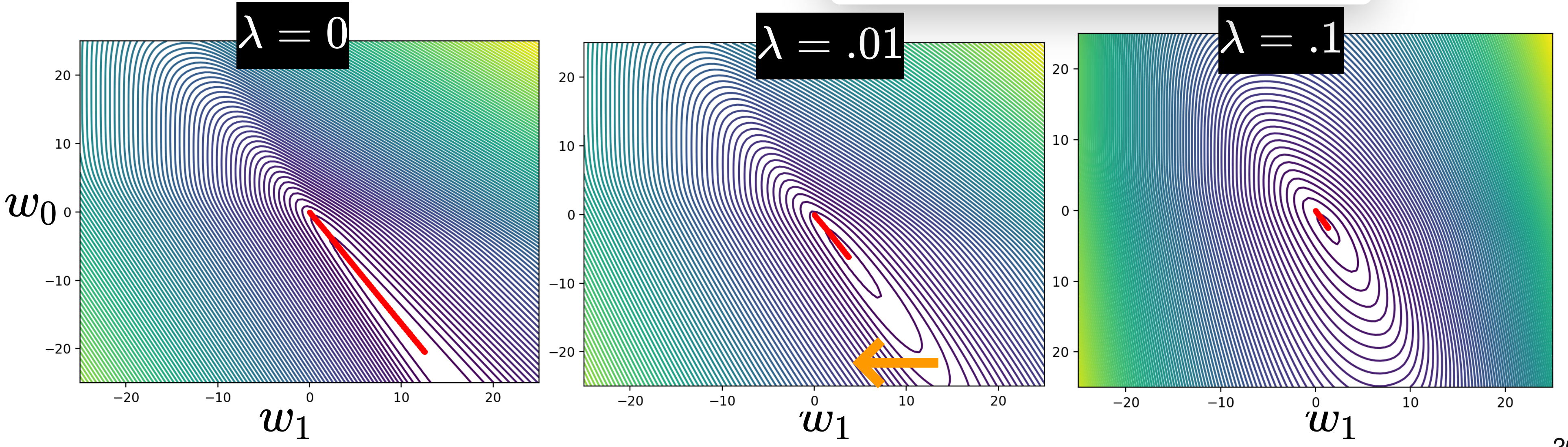
L2 penalty makes the optimization easier too!

note that the optimal w_1 **shrinks**

example for **logistic regression**

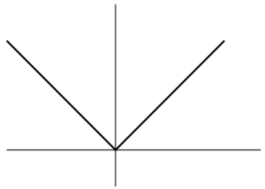
```
def gradient(x, y, w, lambdaa):
    N,D = x.shape
    yh = logistic(np.dot(x, w))
    grad = np.dot(x.T, yh - y) / N
    grad[1:] += lambdaa * w[1:]
    return grad
```

weight decay



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

Sub-derivatives



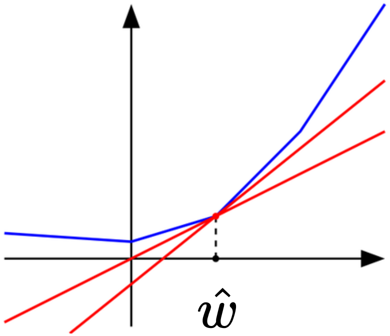
L1 penalty is no longer smooth or differentiable (at 0)

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



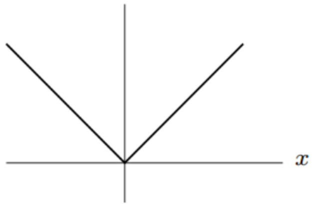
another expression for sub-differential

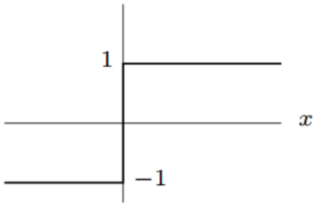
$$\partial f(\hat{w}) = \{g \in \mathbb{R} |\ f(w) > f(\hat{w}) + g(w - \hat{w})\}$$

optional

Subgradient

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





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

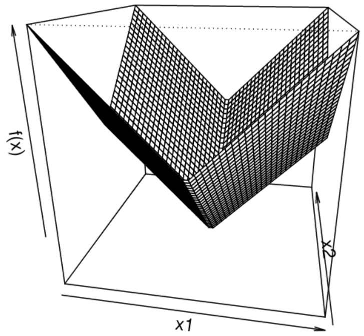
$$\partial f(w \neq 0) = \{\text{sign}(w)\}$$

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

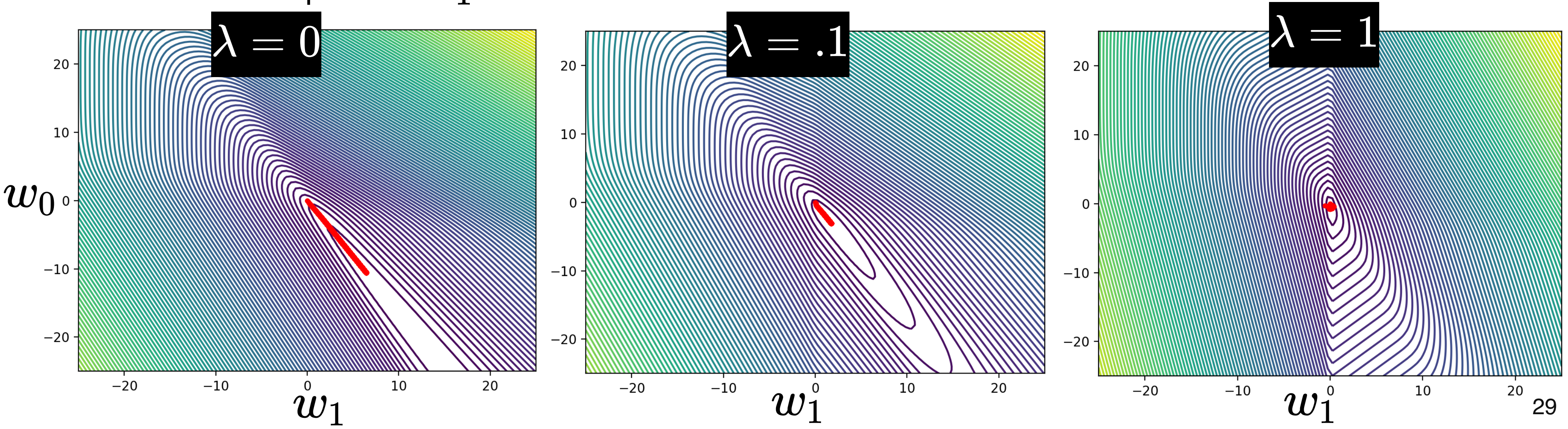
subgradient method for L1-regularized logistic regression

do not penalize the bias w_0

using diminishing learning rate

```
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
```

note that the optimal w_1 **becomes 0**



Regularization serves many purposes

$$\underset{D \times 1}{w^*} = (\underset{D \times N}{X^\top} \underset{N \times D}{X})^{-1} X^\top \underset{N \times 1}{y}$$

what if $X^\top X$ 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!

Summary

- complex models can overfit to training data
- regularization avoids this by penalizing model complexity
  - L1 & L2 regularization
  - probabilistic interpretation: different priors on weights
  - L1 produces sparse solutions (useful for feature selection)