CPSC 340: Machine Learning and Data Mining

More Regularization

Spring 2022 (2021W2)

Last Time: L2-Regularization

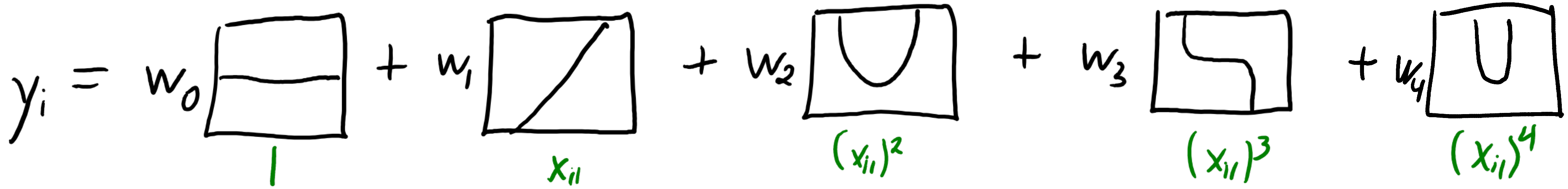
- We discussed regularization:
  - Adding a continuous penalty on the model complexity:

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

  - Best parameter λ almost always leads to improved test error.
    - L2-regularized least squares is also known as "ridge regression".
    - Can be solved as a linear system like least squares.
  - Numerous other benefits:
    - Solution is unique, less sensitive to data, gradient descent converges faster.

Parametric vs. Non-Parametric Transforms

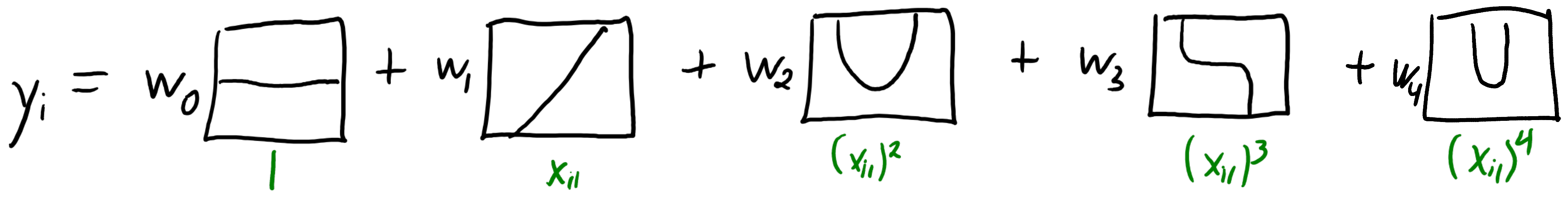
- We’ve been using linear models with polynomial bases:



- But polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.

Parametric vs. Non-Parametric Transforms

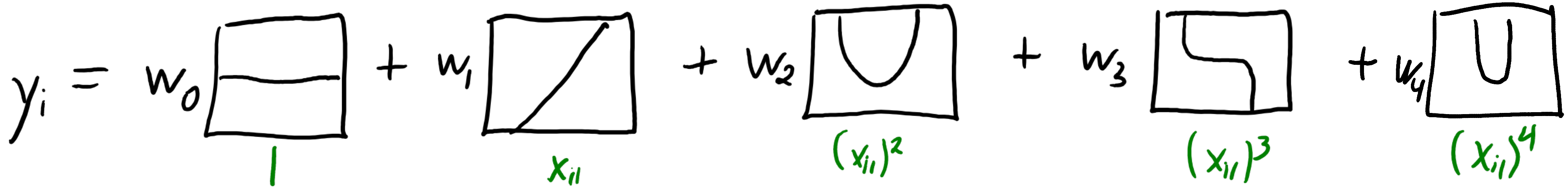
- We've been using linear models with polynomial bases:



- But polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.
  - If we use the wrong basis, our accuracy is limited even with lots of data.
  - But the right basis may not be obvious.

Parametric vs. Non-Parametric Transforms

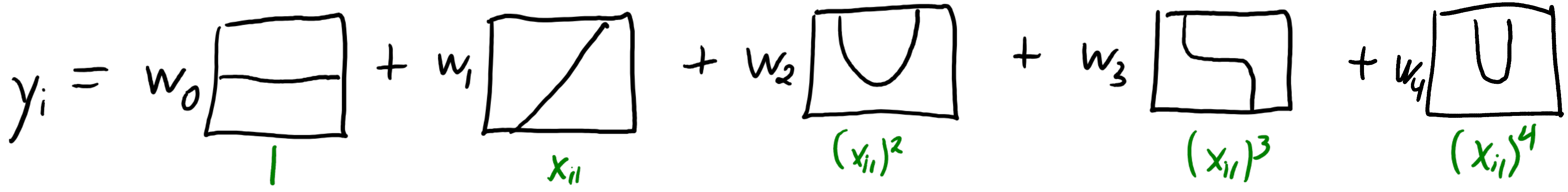
- We've been using linear models with polynomial bases:



- Alternative is non-parametric bases:
  - Size of basis (number of features) grows with 'n'.

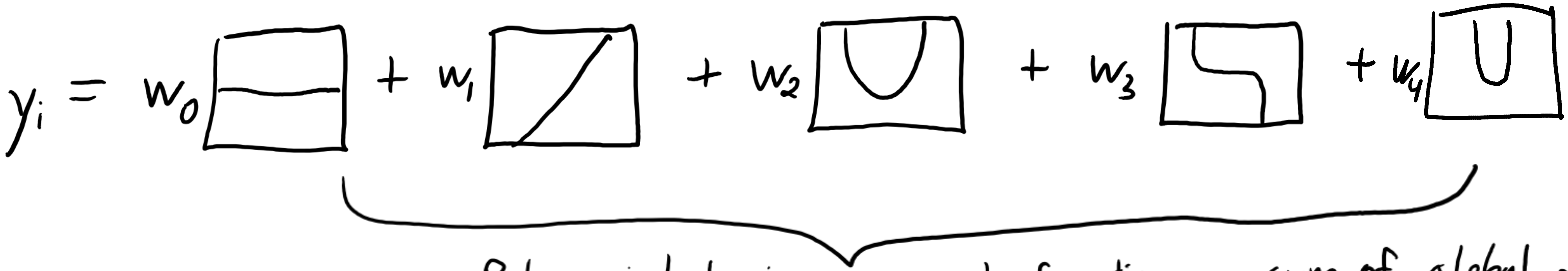
Parametric vs. Non-Parametric Transforms

- We've been using linear models with polynomial bases:



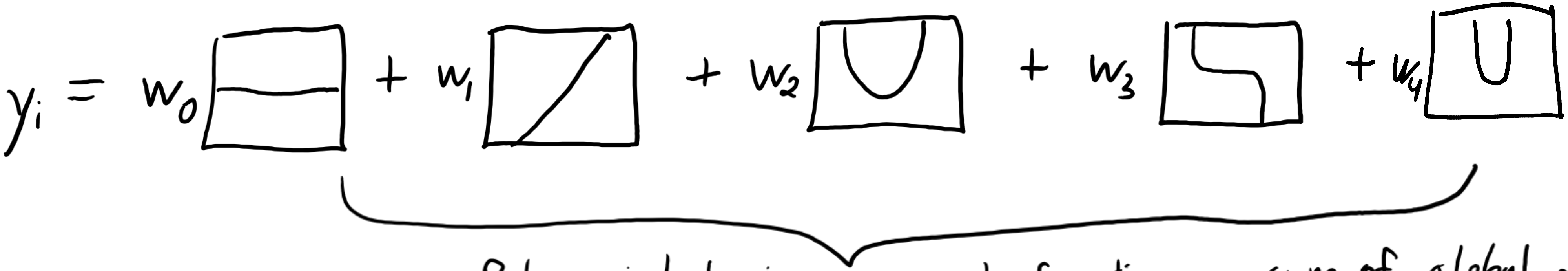
- Alternative is non-parametric bases:
  - Size of basis (number of features) grows with 'n'.
  - Model gets more complicated as you get more data.
  - Can model complicated functions where you don't know the right basis.
    - With enough data.
  - Classic example is "Gaussian RBFs" ("Gaussian" == "normal distribution").

Gaussian RBFs: A Sum of “bumps”

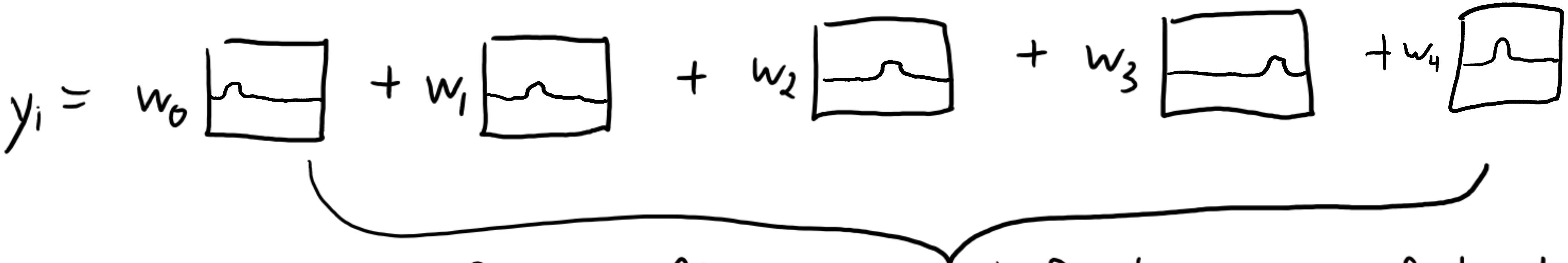


Polynomial basis represents function as sum of <u>global</u> polynomials.

Gaussian RBFs: A Sum of “bumps”

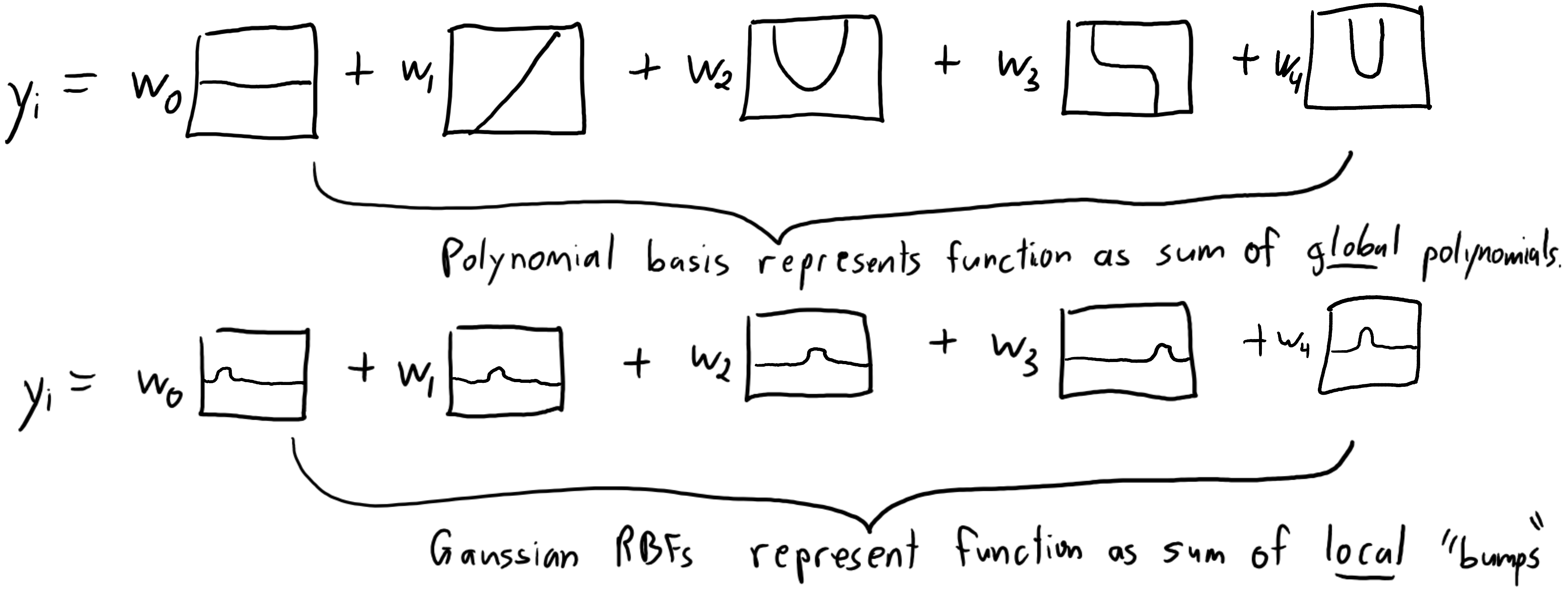


Polynomial basis represents function as sum of global polynomials.



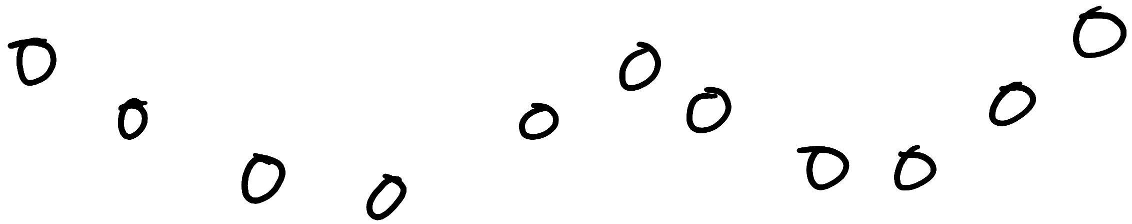
Gaussian RBFs represent function as sum of local “bumps”

Gaussian RBFs: A Sum of "bumps"

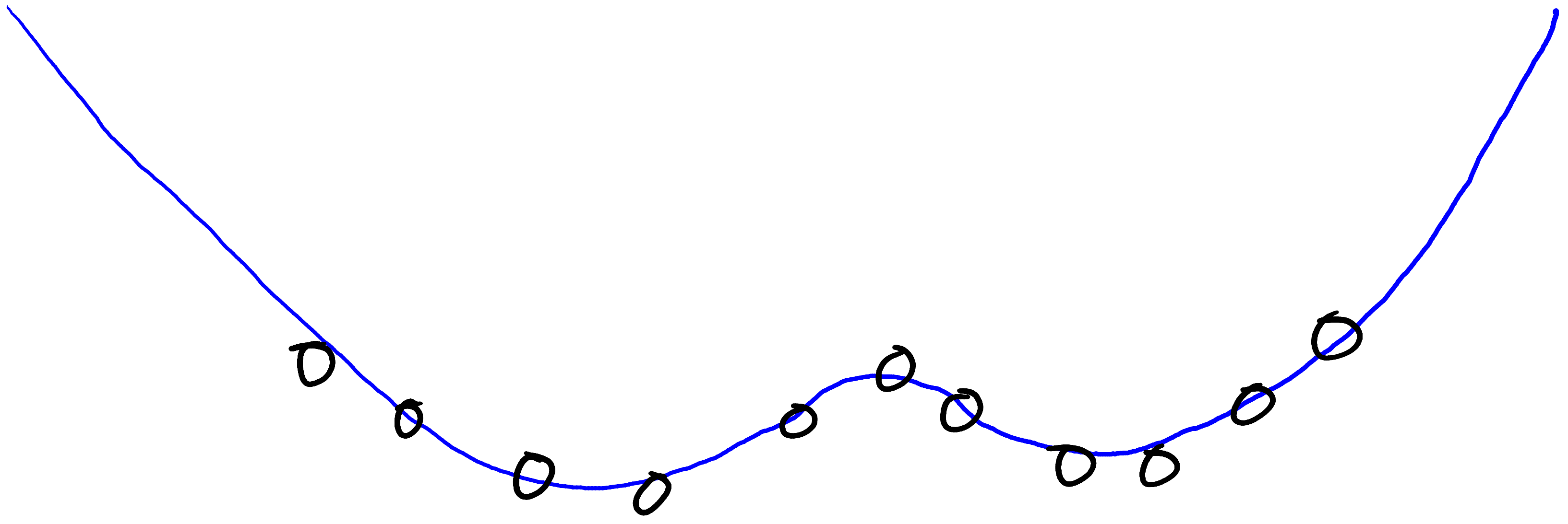


- Gaussian RBFs are universal approximators (on compact subets of $\mathbb{R}^d$)
  - Enough bumps can approximate any continuous function to arbitrary precision.
  - Achieve optimal test error as 'n' goes to infinity.

Gaussian RBFs: A Sum of “Bumps”

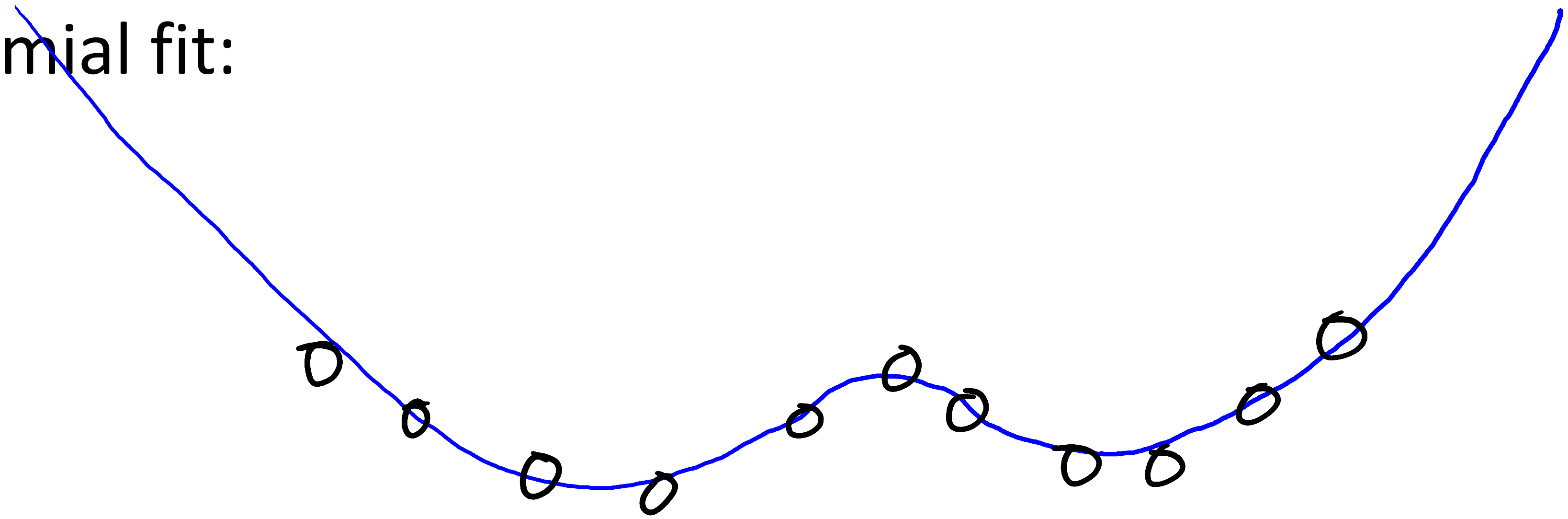


Gaussian RBFs: A Sum of “Bumps”

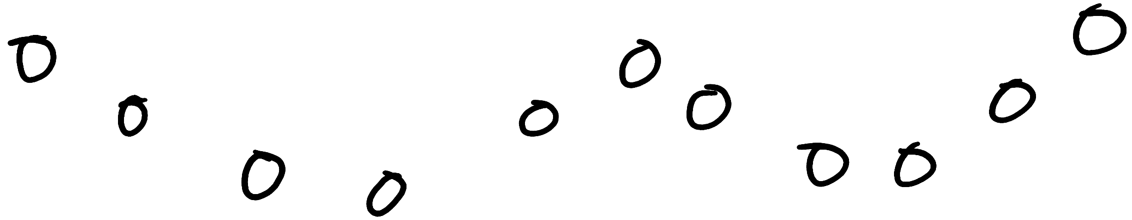


Gaussian RBFs: A Sum of "Bumps"

- Polynomial fit:

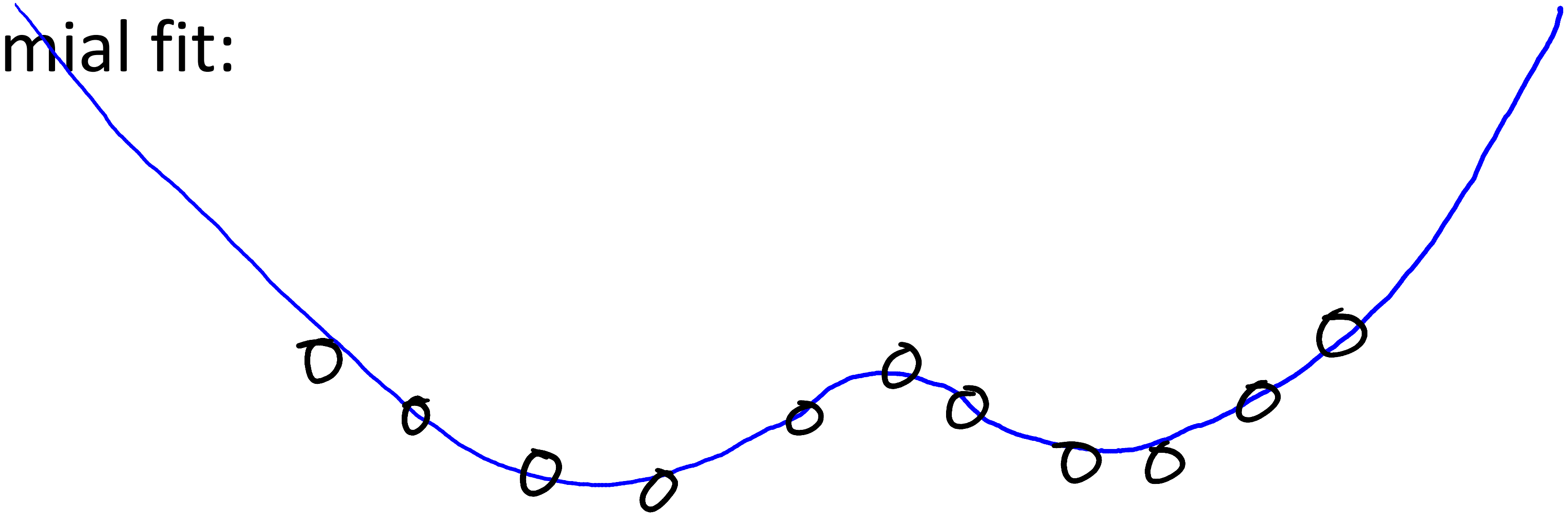


- Constructing a function from bumps ("smooth histogram"):

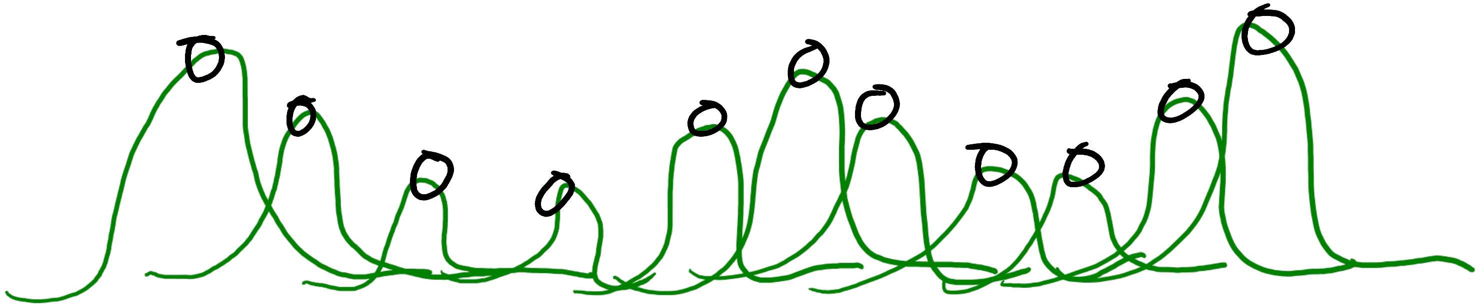


Gaussian RBFs: A Sum of “Bumps”

- Polynomial fit:

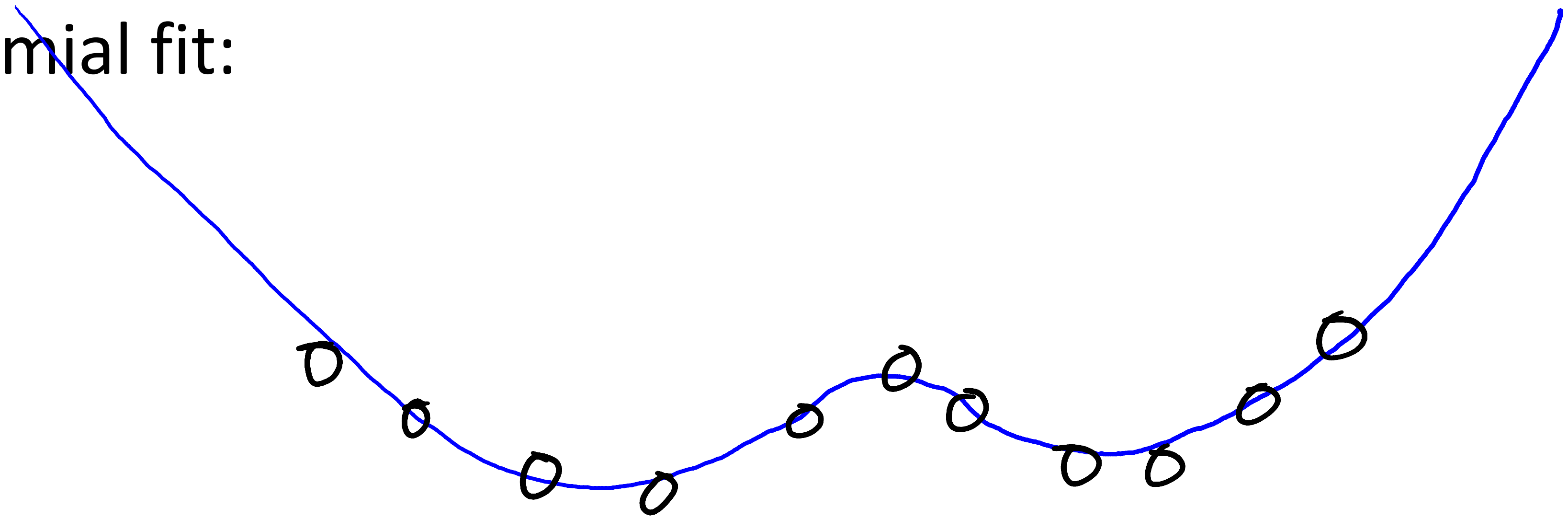


- Constructing a function from bumps (“smooth histogram”):

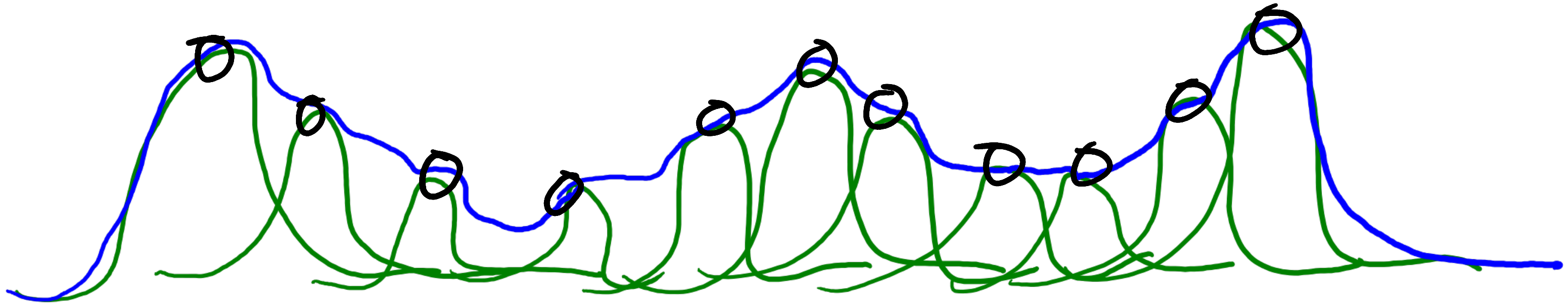


Gaussian RBFs: A Sum of “Bumps”

- Polynomial fit:

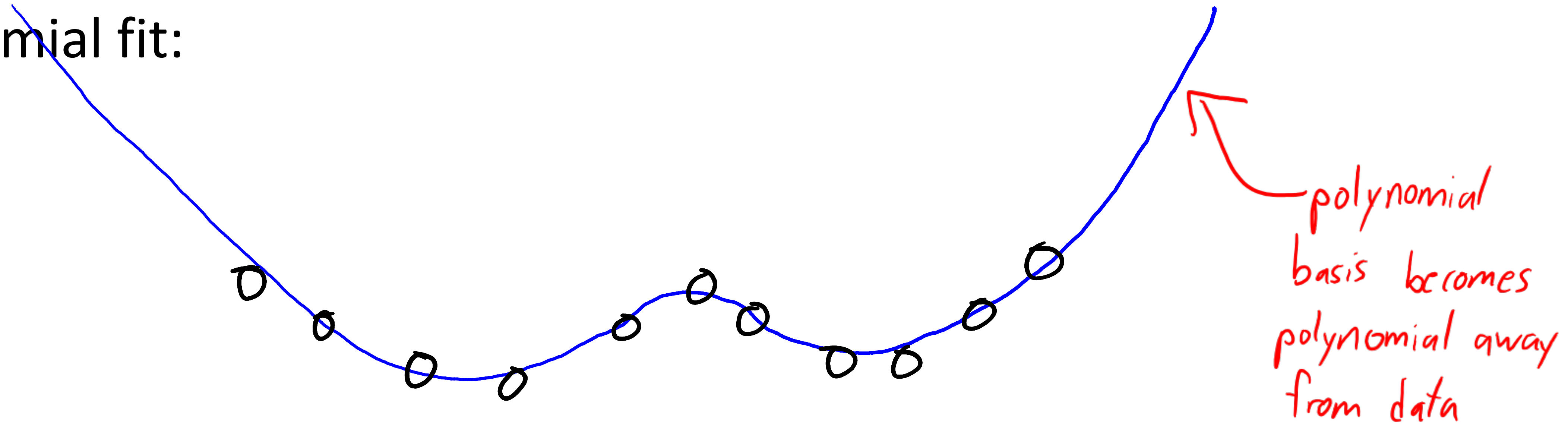


- Constructing a function from bumps (“smooth histogram”):

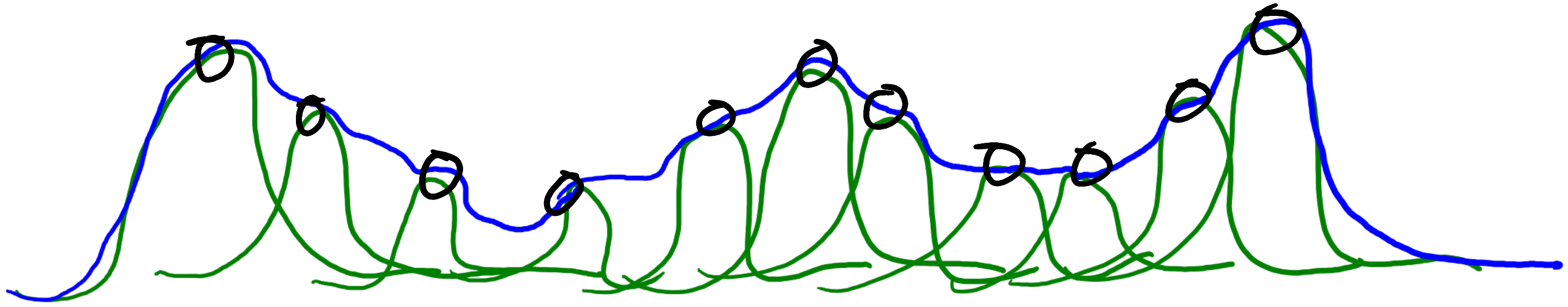


Gaussian RBFs: A Sum of "Bumps"

- Polynomial fit:

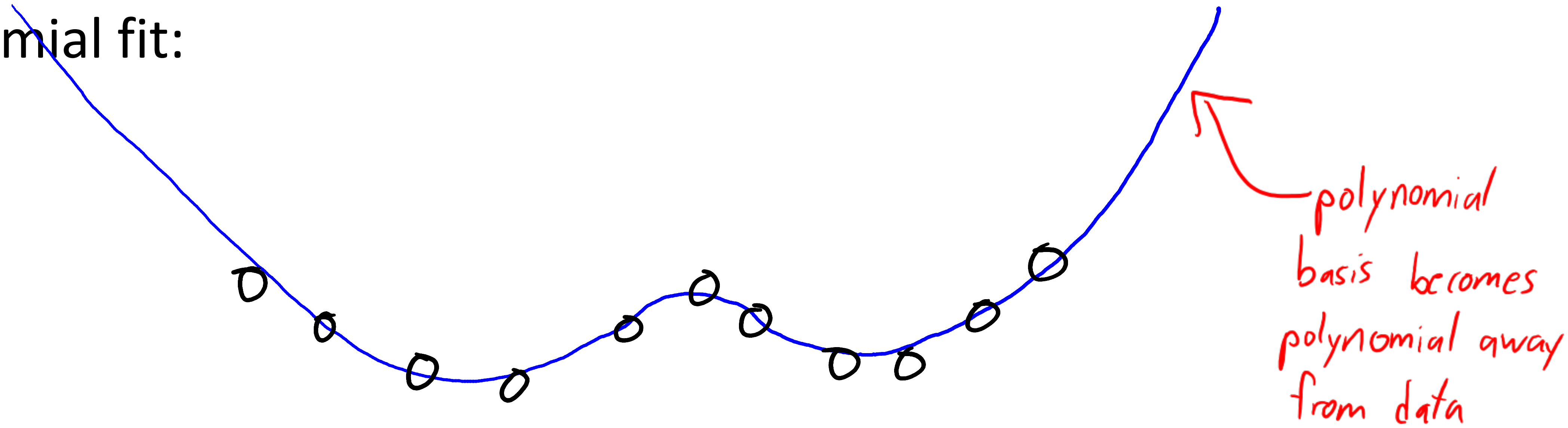


- Constructing a function from bumps ("smooth histogram"):

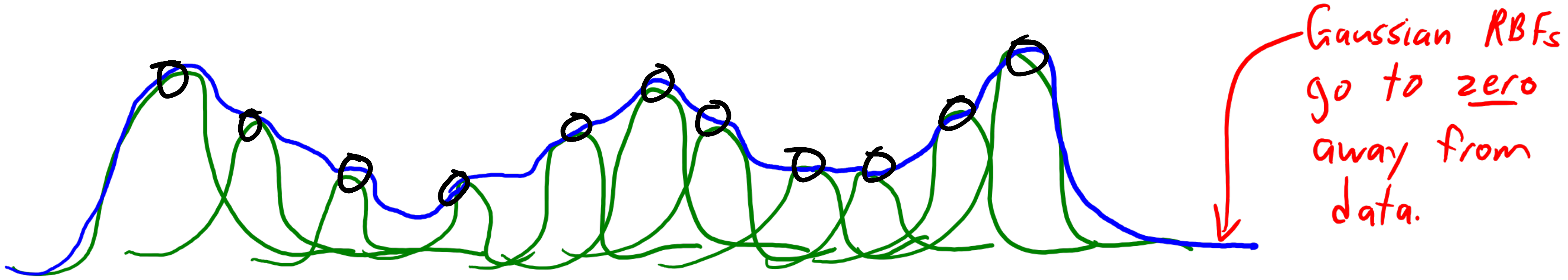


Gaussian RBFs: A Sum of "Bumps"

- Polynomial fit:

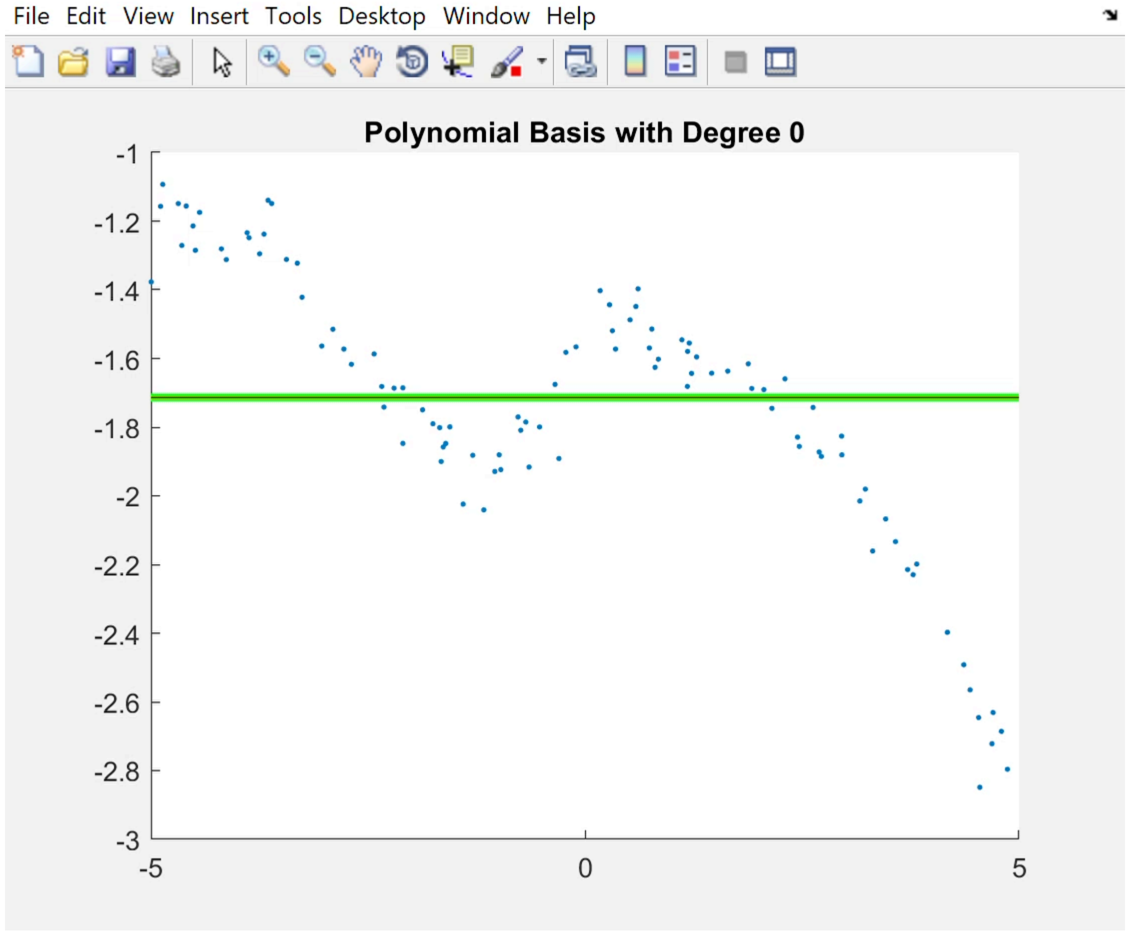


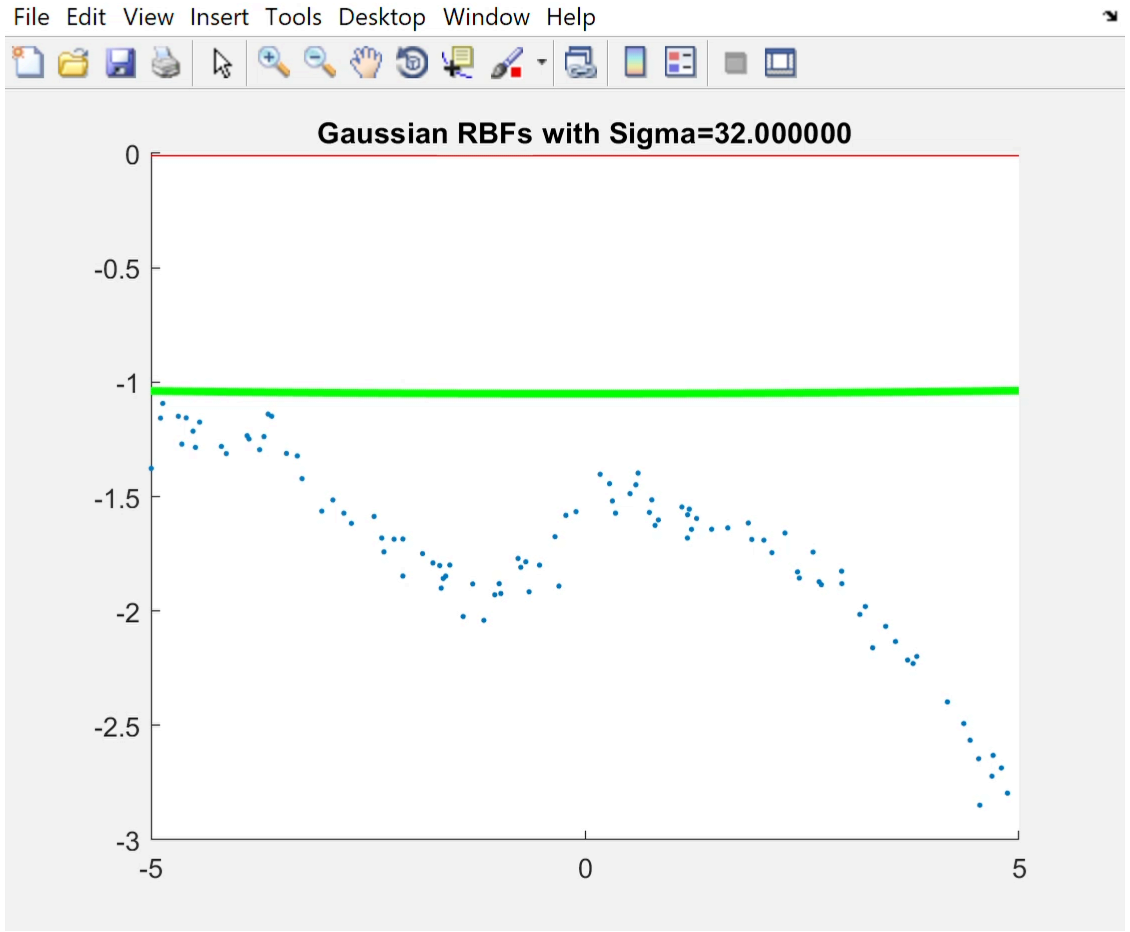
- Constructing a function from bumps ("smooth histogram"):



Gaussian RBFs: A Sum of "Bumps"

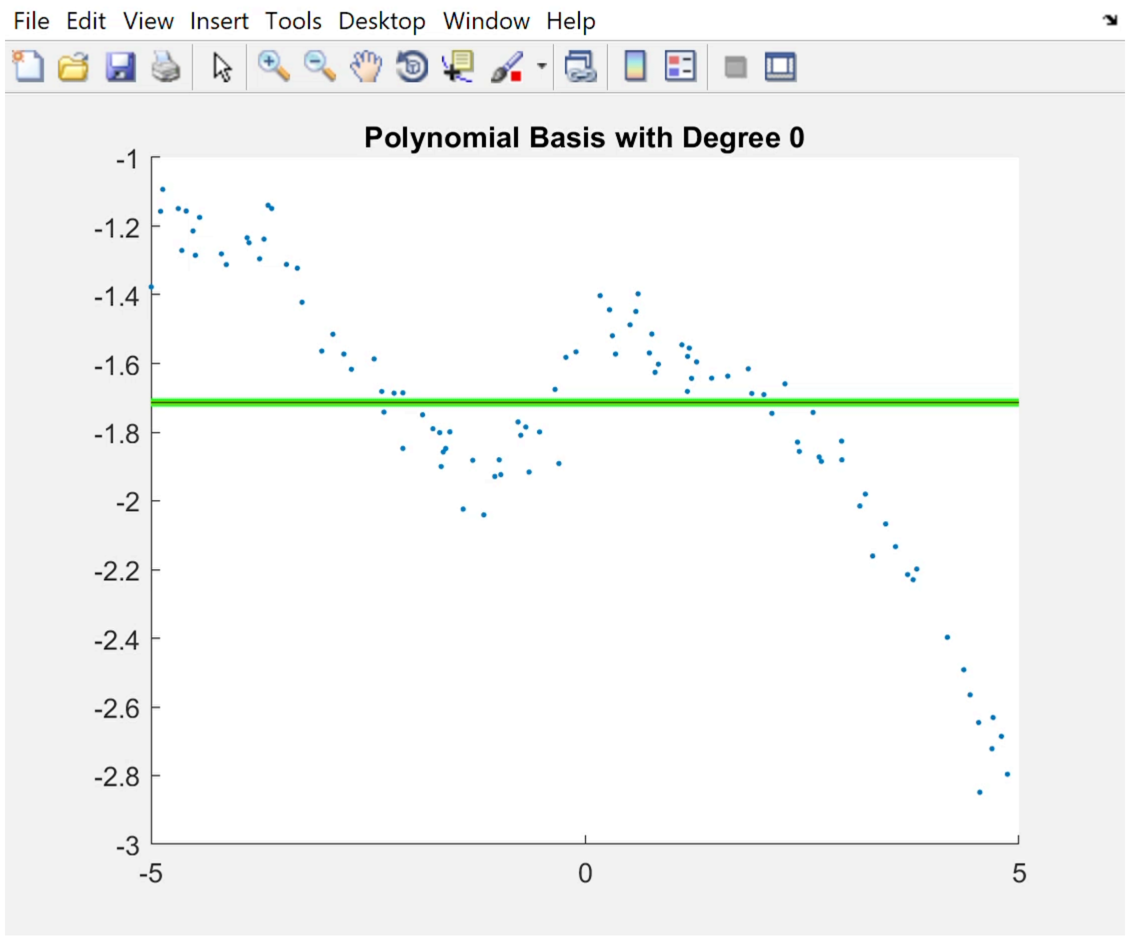
- More-realistic version (green is regression line, red is each basis):

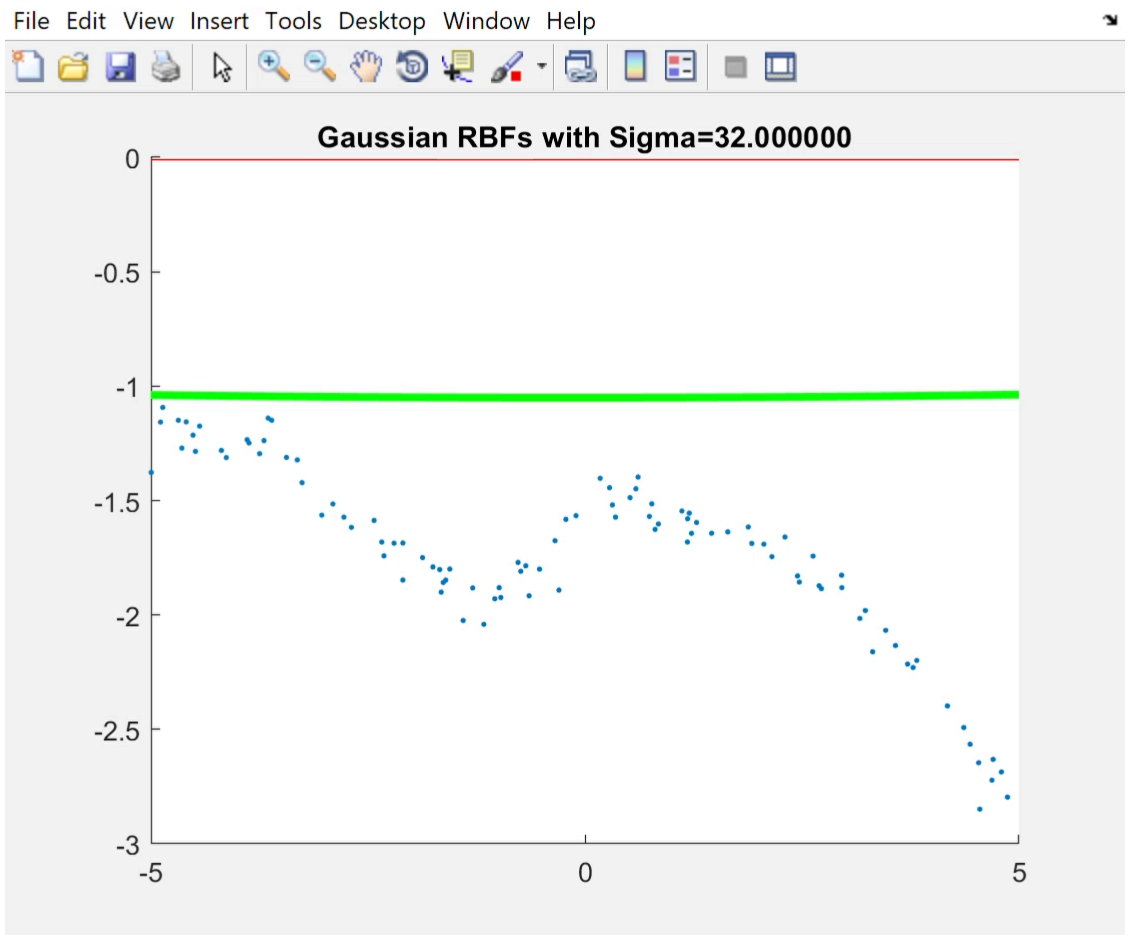




Gaussian RBFs: A Sum of “Bumps”

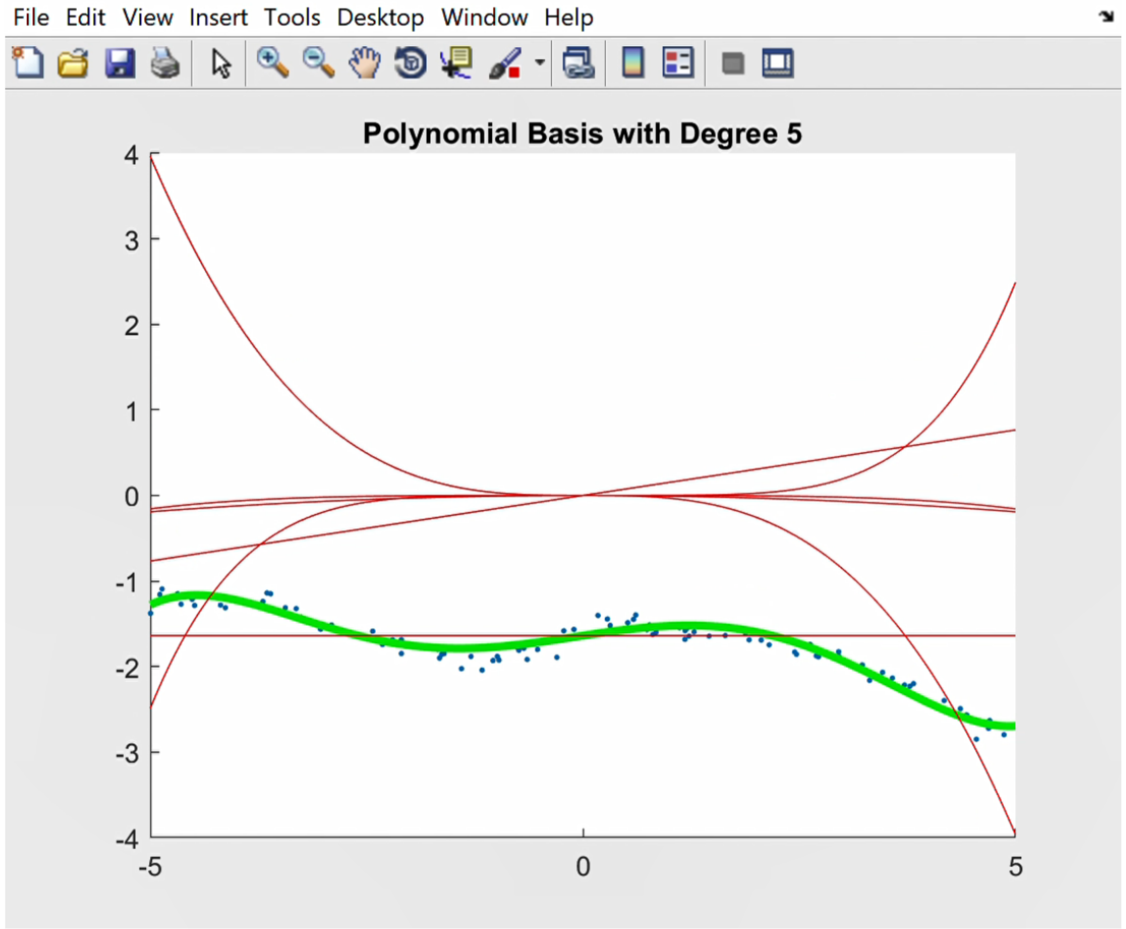
- More-realistic version (green is regression line, red is each basis):

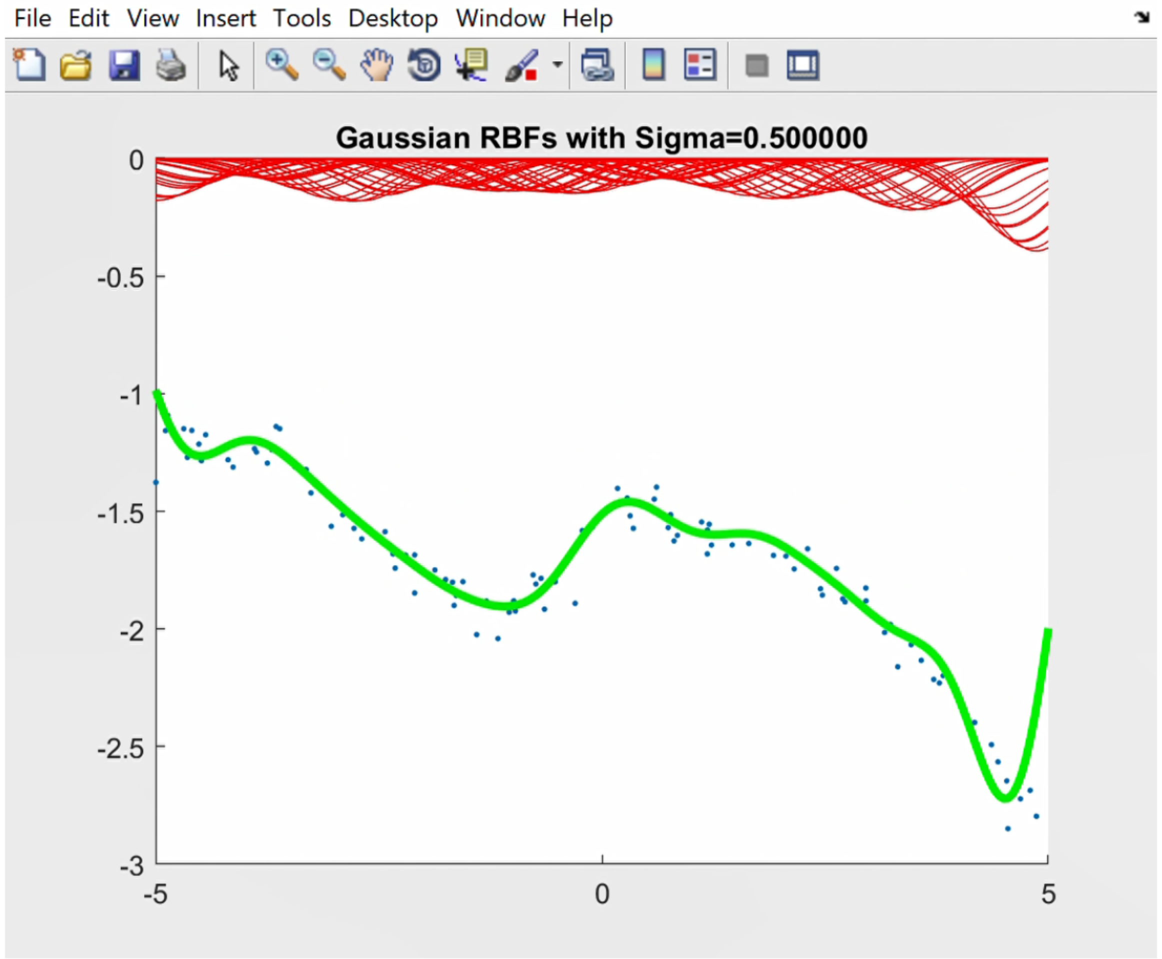




Gaussian RBFs: A Sum of "Bumps"

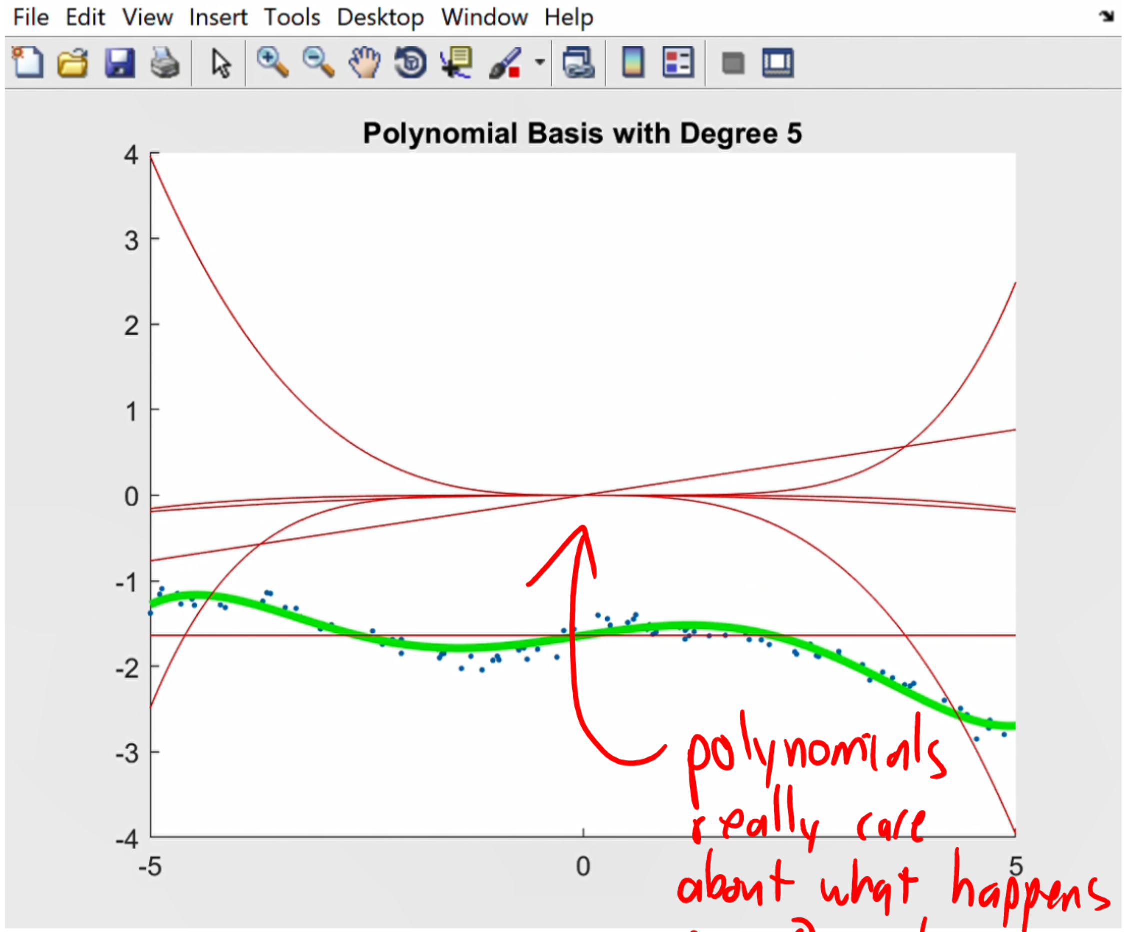
- More-realistic version (green is regression line, red is each basis):



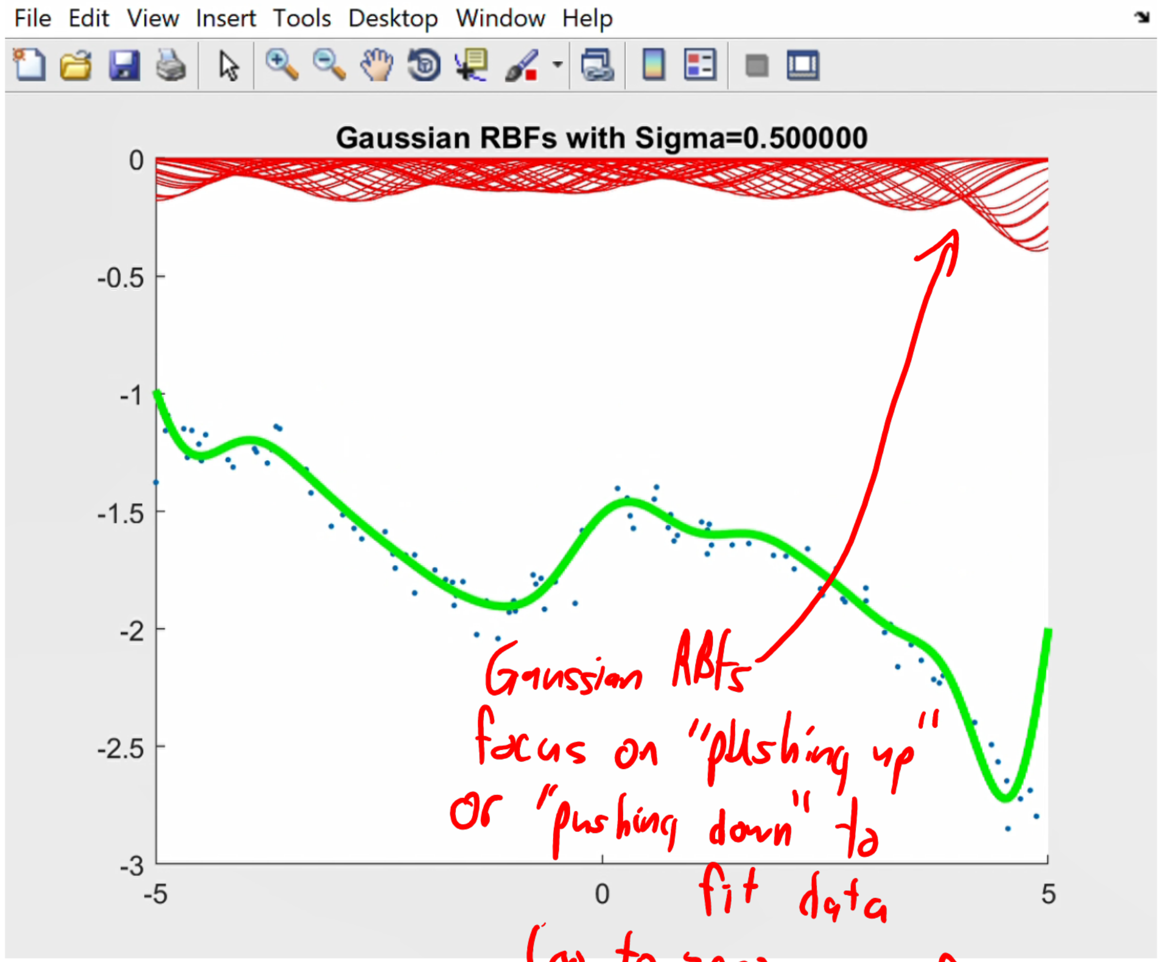


Gaussian RBFs: A Sum of "Bumps"

- More-realistic version (green is regression line, red is each basis):



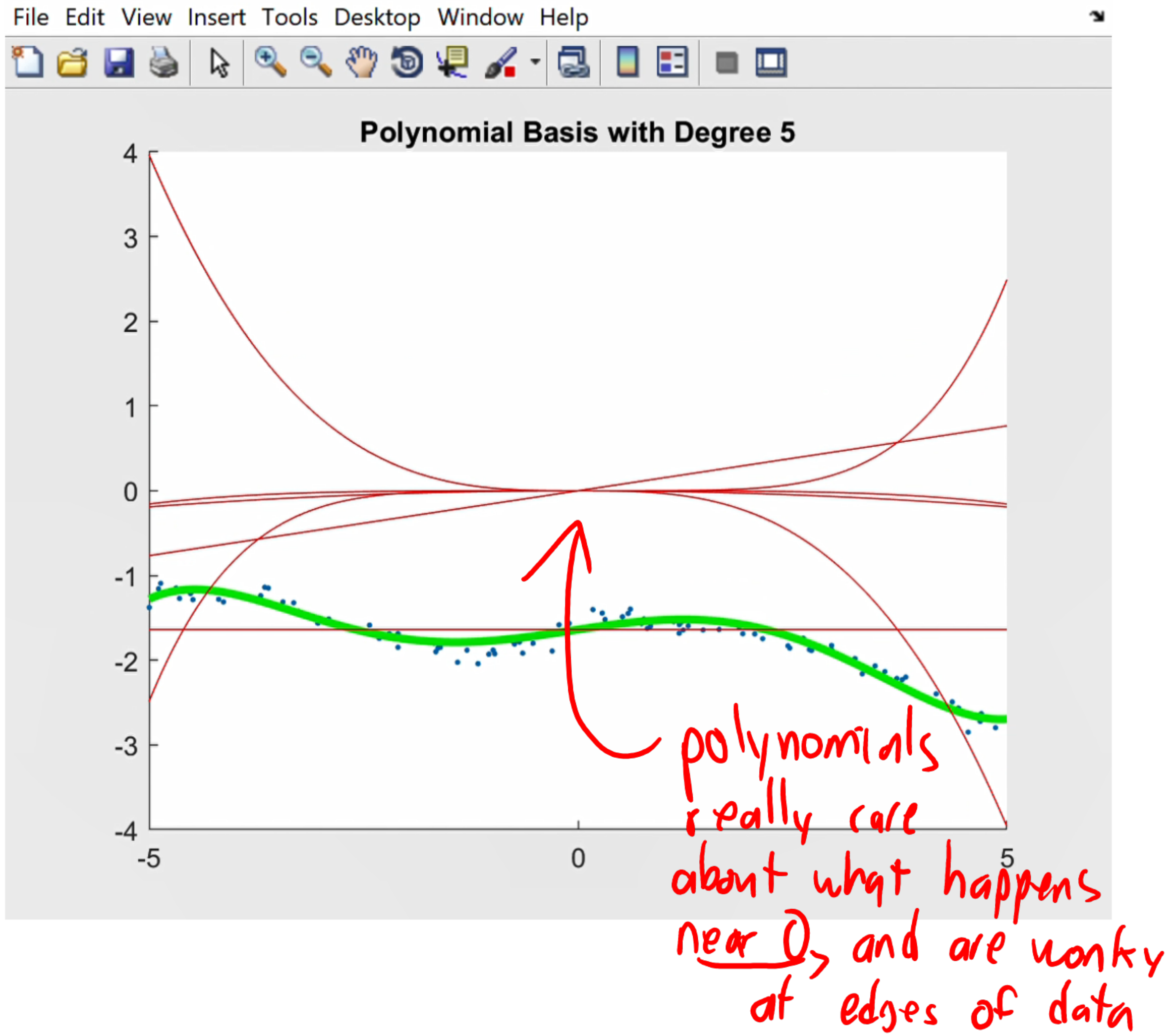
polynomials really care about what happens near 0, and are wonky at edges of data

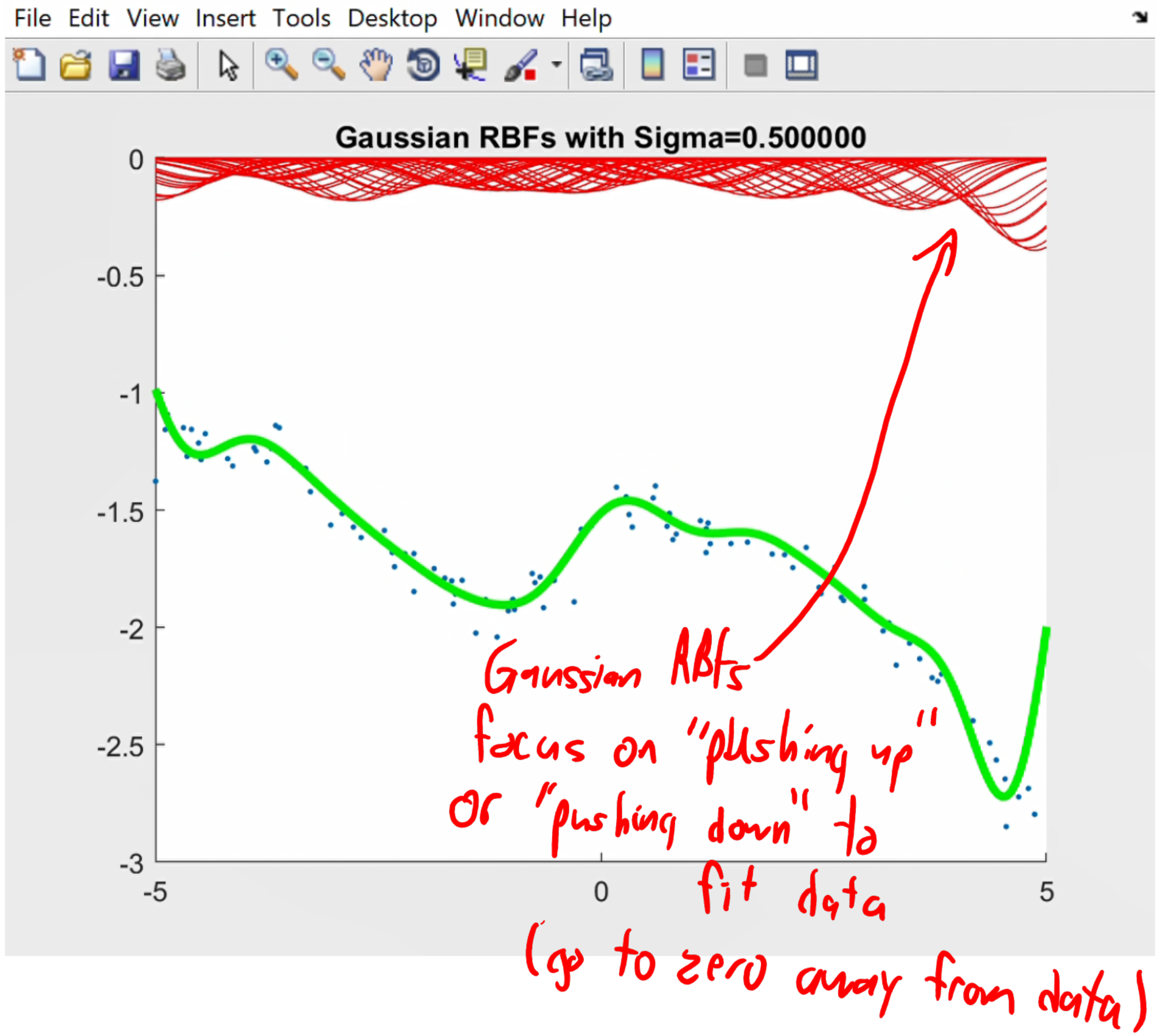


Gaussian RBFs focus on "pushing up" or "pushing down" to fit data (go to zero away from data)

Gaussian RBFs: A Sum of “Bumps”

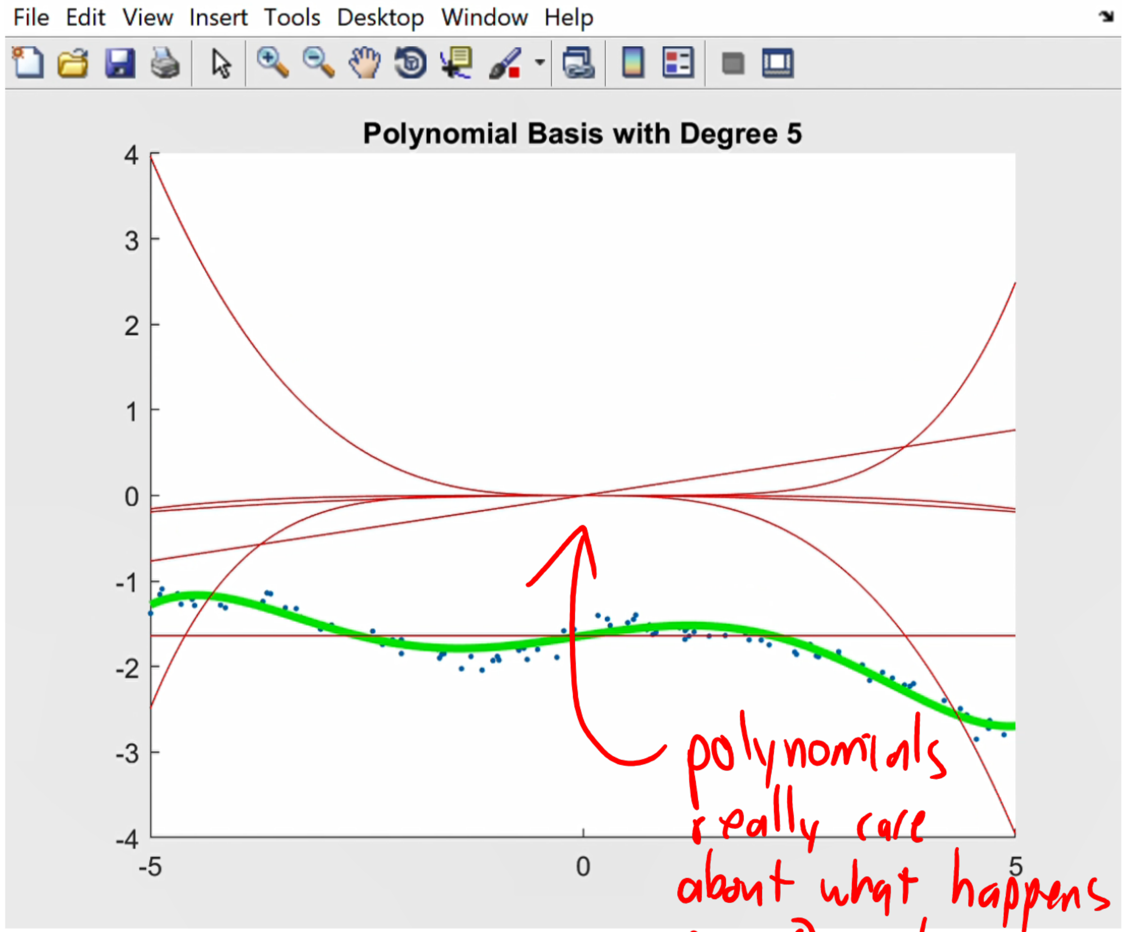
- More-realistic version (green is regression line, red is each basis):



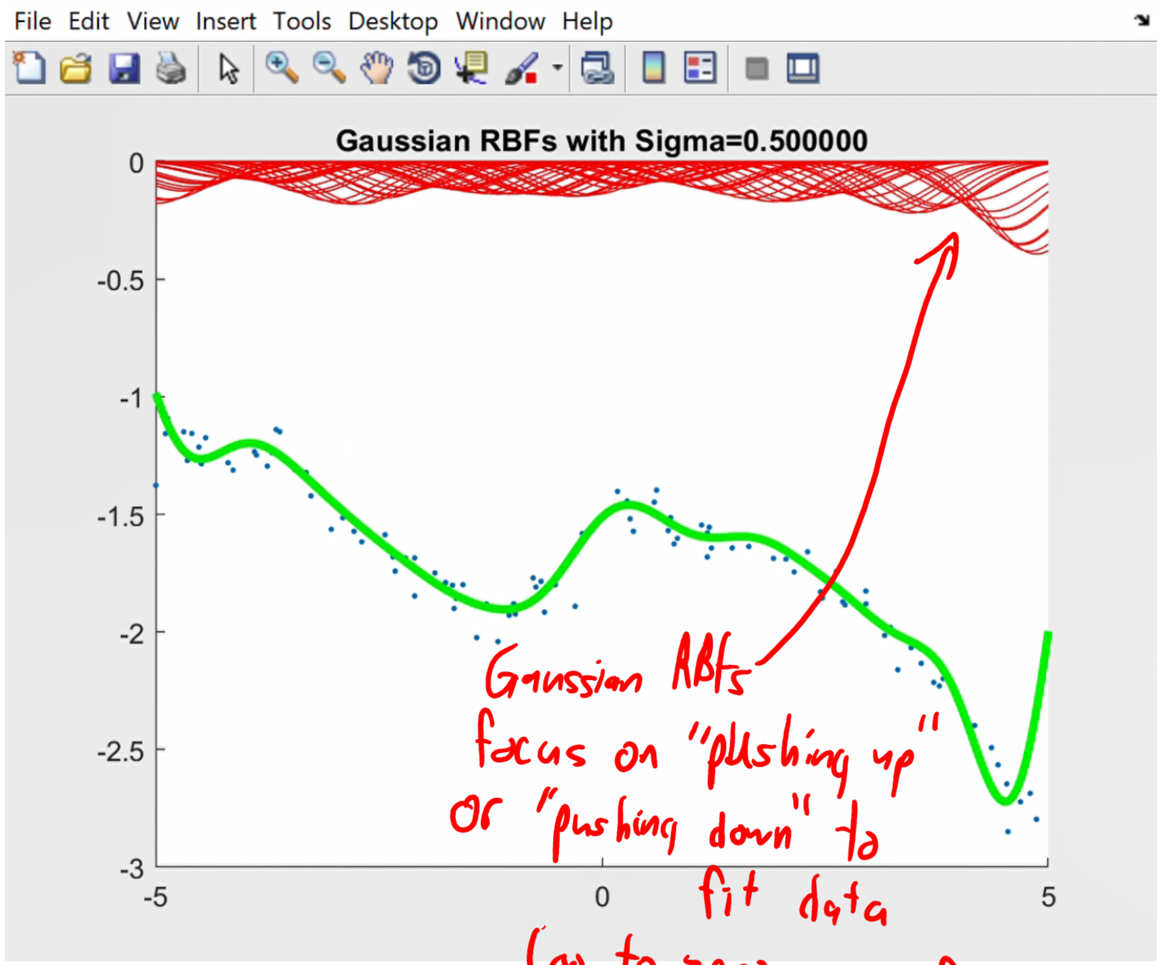


Gaussian RBFs: A Sum of “Bumps”

- More-realistic version (green is regression line, red is each basis):



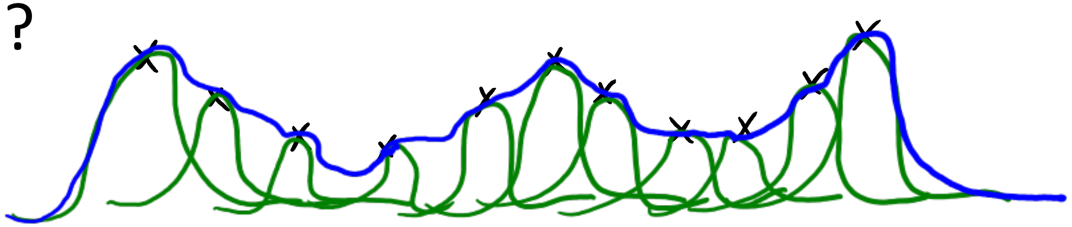
polynomials really care about what happens near 0, and are wonky at edges of data



Gaussian RBFs focus on "pushing up" or "pushing down" to fit data (go to zero away from data)

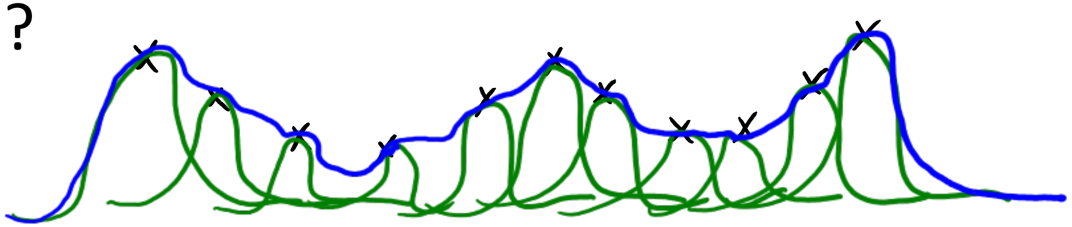
Gaussian RBF Parameters

- Some obvious questions:
    1. How many bumps should we use?
    2. Where should the bumps be centered?
    3. How high should the bumps go?
    4. How wide should the bumps be?



Gaussian RBF Parameters

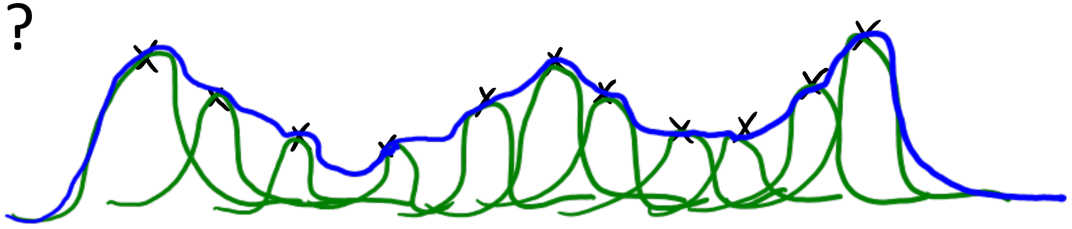
- Some obvious questions:
  1. How many bumps should we use?
  2. Where should the bumps be centered?
  3. How high should the bumps go?
  4. How wide should the bumps be?



- The usual answers:
  1. We use 'n' bumps (non-parametric basis).

Gaussian RBF Parameters

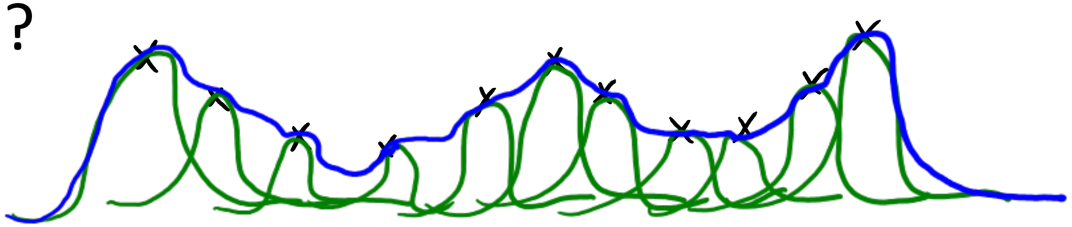
- Some obvious questions:
  1. How many bumps should we use?
  2. Where should the bumps be centered?
  3. How high should the bumps go?
  4. How wide should the bumps be?



- The usual answers:
  1. We use 'n' bumps (non-parametric basis).
  2. Each bump is centered on one training example x_i.

Gaussian RBF Parameters

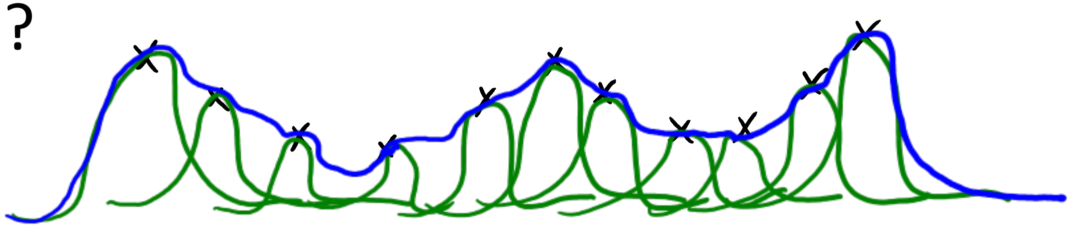
- Some obvious questions:
  1. How many bumps should we use?
  2. Where should the bumps be centered?
  3. How high should the bumps go?
  4. How wide should the bumps be?



- The usual answers:
  1. We use 'n' bumps (non-parametric basis).
  2. Each bump is centered on one training example x_i.
  3. Fitting regression weights 'w' gives us the heights (and signs).

Gaussian RBF Parameters

- Some obvious questions:
  1. How many bumps should we use?
  2. Where should the bumps be centered?
  3. How high should the bumps go?
  4. How wide should the bumps be?



- The usual answers:
  1. We use ‘n’ bumps (non-parametric basis).
  2. Each bump is centered on one training example x_i.
  3. Fitting regression weights ‘w’ gives us the heights (and signs).
  4. The width is a hyper-parameter (narrow bumps == complicated model).

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{in}}_{\text{'d' features}})$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{in}}_{\text{'d' features}})$ with $z_i = (g(\|x_i - x_1\|)$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{in}}_{\text{'d' features}})$ with $z_i = (g(\|x_i - x_1\|), g(\|x_i - x_2\|)$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{in}}_{\text{'d' features}})$ with $z_i = (\underbrace{g(\|x_i - x_1\|), g(\|x_i - x_2\|), \dots, g(\|x_i - x_n\|)}_{\text{'n' features}})$

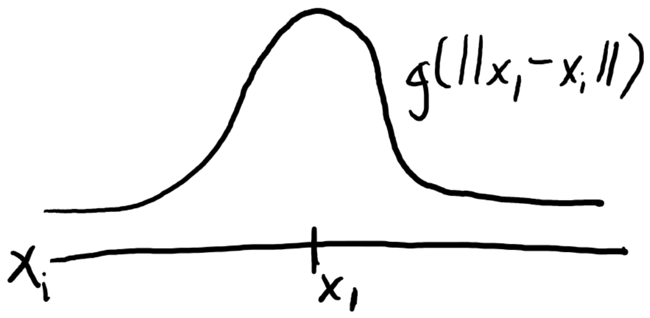
  - Have ‘n’ features, with feature ‘j’ depending on distance to example ‘i’.
    - Typically the feature will decrease as the distance increases:

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

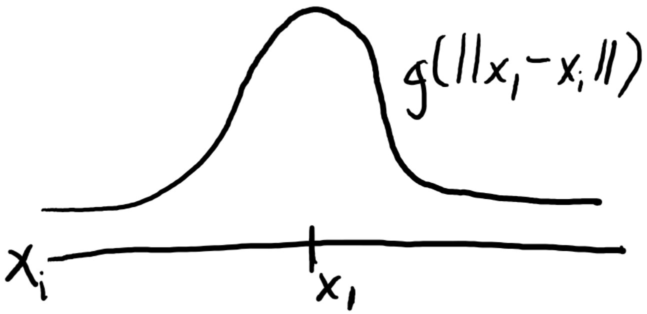
Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{in}}_{\text{'d' features}})$ with $z_i = (\underbrace{g(\|x_i - x_1\|), g(\|x_i - x_2\|), \dots, g(\|x_i - x_n\|)}_{\text{'n' features}})$

  - Have ‘n’ features, with feature ‘j’ depending on distance to example ‘i’.
    - Typically the feature will decrease as the distance increases:



Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - Most common choice of 'g' is Gaussian RBF:

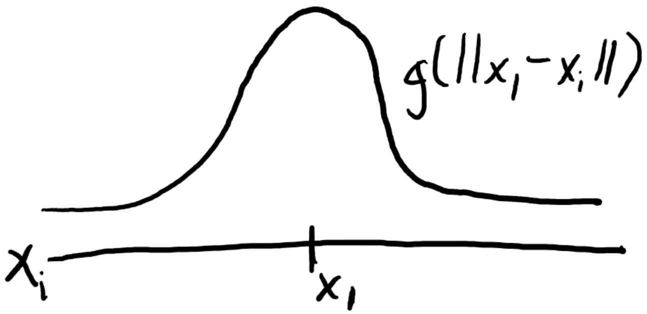


bonus!

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - Most common choice of 'g' is Gaussian RBF:

$$g(\varepsilon) = \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

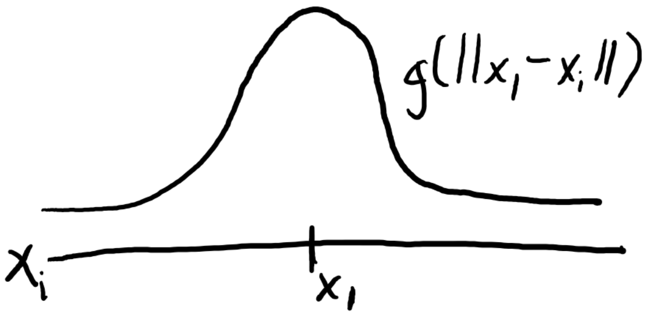


bonus!

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - Most common choice of 'g' is Gaussian RBF:

$$g(\varepsilon) = \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$



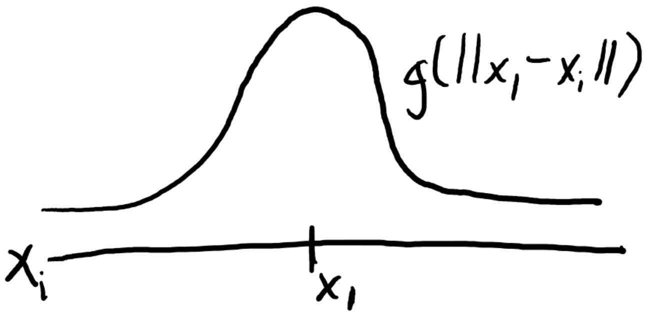
    - Variance σ^2 is a hyper-parameter controlling "width".
      - This affects fundamental trade-off (set it using a validation set).

bonus!

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - Most common choice of 'g' is Gaussian RBF:

$$g(\varepsilon) = \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$



    - Variance σ^2 is a hyper-parameter controlling "width".
      - This affects fundamental trade-off (set it using a validation set).

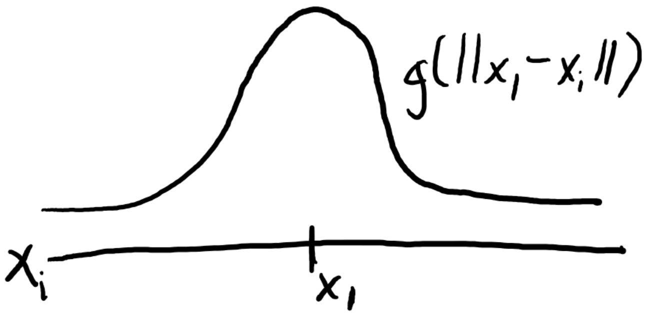
- Why don't we have $\sqrt{2\pi\sigma}$ in the above formula?

bonus!

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - Most common choice of 'g' is Gaussian RBF:

$$g(\varepsilon) = \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$



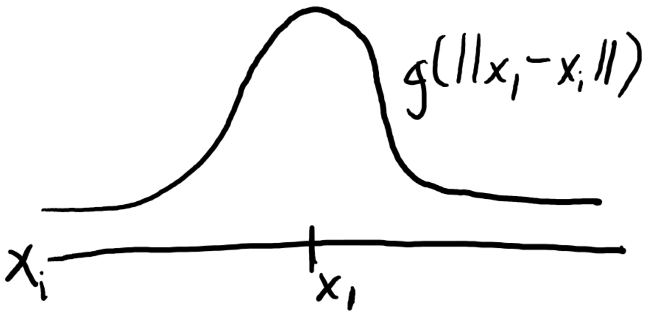
- Variance σ^2 is a hyper-parameter controlling "width".
  - This affects fundamental trade-off (set it using a validation set).

bonus!

- Why don't we have $\sqrt{2\pi\sigma}$ in the above formula?
  - If you don't regularize it does not matter:
    - If 'v' is least squares solution with features z_i, then $(\sqrt{2\pi\sigma})v$ is solution with features $(1/\sqrt{2\pi\sigma})z_i$.
    - So you get the same predictions (least squares is invariant to scaling of features).

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - Most common choice of 'g' is Gaussian RBF:



$$g(\varepsilon) = \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

- Variance σ^2 is a hyper-parameter controlling "width".
  - This affects fundamental trade-off (set it using a validation set).

bonus!

- Why don't we have $\sqrt{2\pi\sigma}$ in the above formula?
  - If you don't regularize it does not matter:
    - If 'v' is least squares solution with features z_i, then $(\sqrt{2\pi\sigma})$v is solution with features $(1/\sqrt{2\pi\sigma})z_i$.
    - So you get the same predictions (least squares is invariant to scaling of features).
  - If you regularize it "sort of" matters:
    - It changes the effect of a fixed λ.
    - But the regularization path is the same, so if you search for the best λ you get same predictions.

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

$$\text{Replace} \quad X = \underbrace{\begin{bmatrix} \quad & \quad \\ & \\ & \end{bmatrix}}_{d} \Big\} n \qquad \text{by} \qquad Z = \underbrace{\begin{bmatrix} \qquad & \qquad & \qquad \\ & & \\ & & \end{bmatrix}}_{n} \Big\} n$$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

Replace $X = \underbrace{\begin{bmatrix} & \\ & \\ & \end{bmatrix}}_{d} \left.\vphantom{\begin{bmatrix} x \\ x \\ x \end{bmatrix}}\right\} n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & \\ & \\ & \end{bmatrix}}_{n} \left.\vphantom{\begin{bmatrix} x \\ x \\ x \end{bmatrix}}\right\} n$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

Replace $X = \underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{d} \Big\} n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \\ & & \\ & & \end{bmatrix}}_{n} \Big\} n$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

$$\text{Replace} \quad X = \underbrace{\begin{bmatrix} & \\ & \\ & \end{bmatrix}}_{d} \Big\} n \qquad \text{by} \qquad Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ & & & \\ & & & \end{bmatrix}}_{n} \Big\} n$$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

Replace $X = \underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{d} \Big\} n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ & & & \end{bmatrix}}_{n} \Big\} n$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

$$\text{Replace } X = \underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{d} \Big\}\, n \quad \text{by} \quad Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}}_{n} \Big\}\, n$$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

Replace $X = \underbrace{\left.\begin{bmatrix} \\ \\ \\ \end{bmatrix}\right\}n}_{d}$ by $Z = \underbrace{\left.\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}\right\}n}_{n}$

To make predictions on $\tilde{X} = \underbrace{\left.\begin{bmatrix} \\ \\ \\ \end{bmatrix}\right\}t}_{d}$ use $\tilde{Z} = \underbrace{\left.\begin{bmatrix} \\ g(\|\tilde{x}_i - x_j\|) \\ \\ \end{bmatrix}\right\}t}_{n}$

Number of "features" is number of training examples.

Gaussian RBFs: Pseudo-Code

Constructing Gaussian RBFs given data 'X' and hyper-parameter σ:

Gaussian RBFs: Pseudo-Code

Constructing Gaussian RBFs given data 'X' and hyper-parameter σ:

$Z = \text{zeros}(n, n)$

Gaussian RBFs: Pseudo-Code

Constructing Gaussian RBFs given data 'X' and hyper-parameter σ:

```
Z = zeros(n,n)
for i1 in 1:n
    for i2 in 1:n
```

Gaussian RBFs: Pseudo-Code

Constructing Gaussian RBFs given data 'X' and hyper-parameter σ:

```
Z = zeros(n,n)
for i1 in 1:n
    for i2 in 1:n
        Z[i1, i2] = exp(-norm(X[i1,:] - X[i2,:])^2 / 2σ^2)
```

Gaussian RBFs: Pseudo-Code

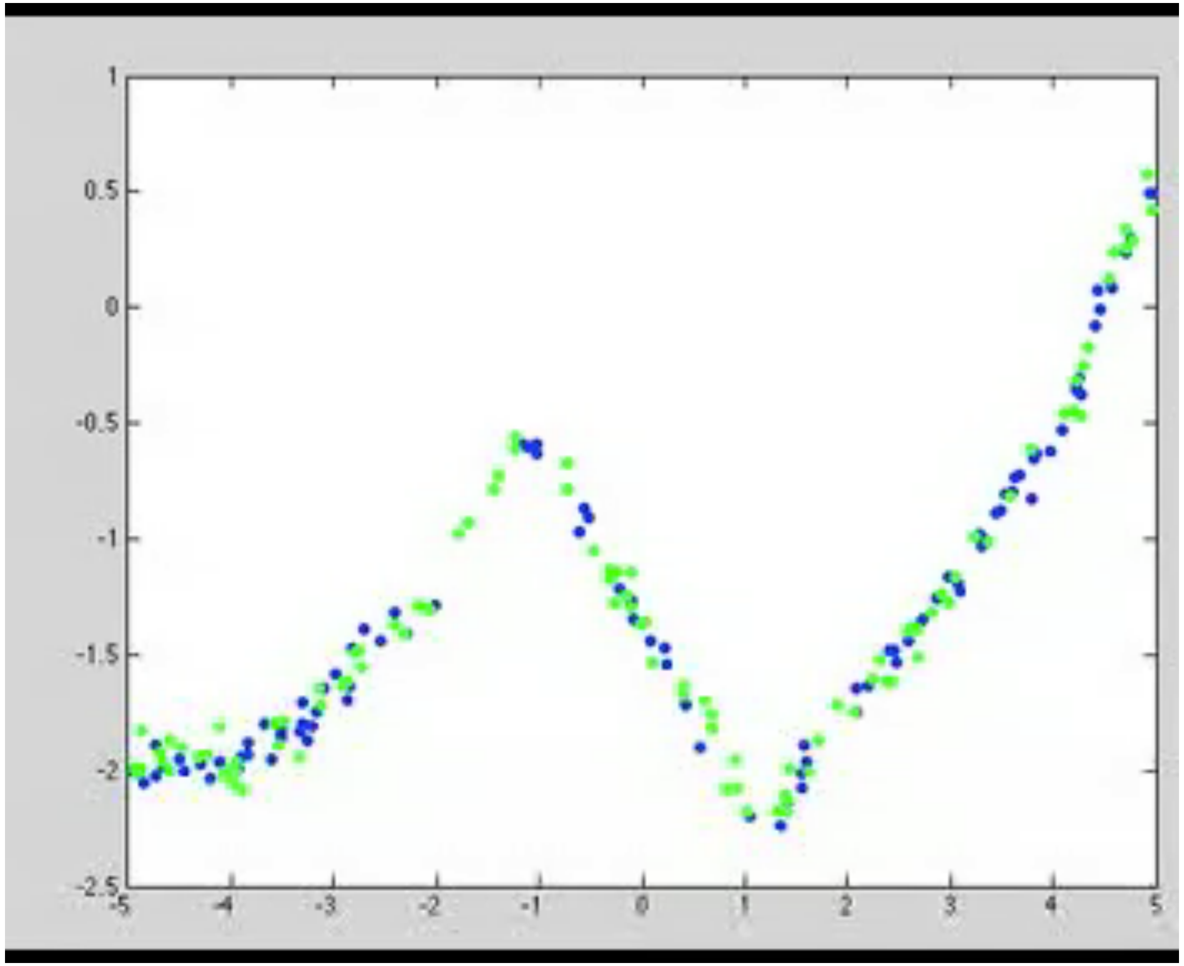
Constructing Gaussian RBFs given data 'X' and hyper-parameter σ:

```
Z = zeros(n,n)
for i1 in 1:n
    for i2 in 1:n
        Z[i1, i2] = exp(-norm(X[i1,:] - X[i2,:])^2 / 2σ^2)
```

With test data $\tilde{X}$: form $\tilde{Z}$ based on distances to training examples.

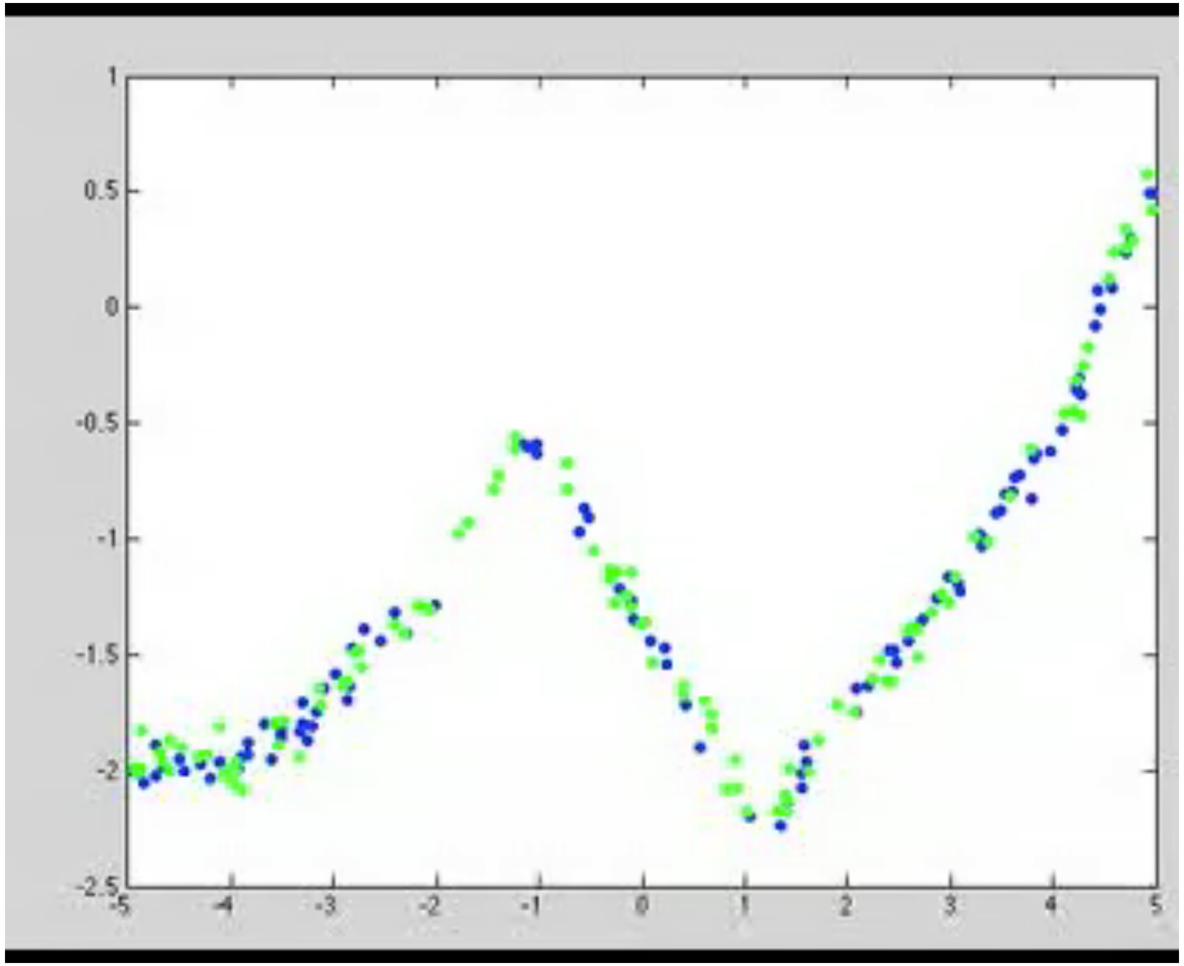
Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



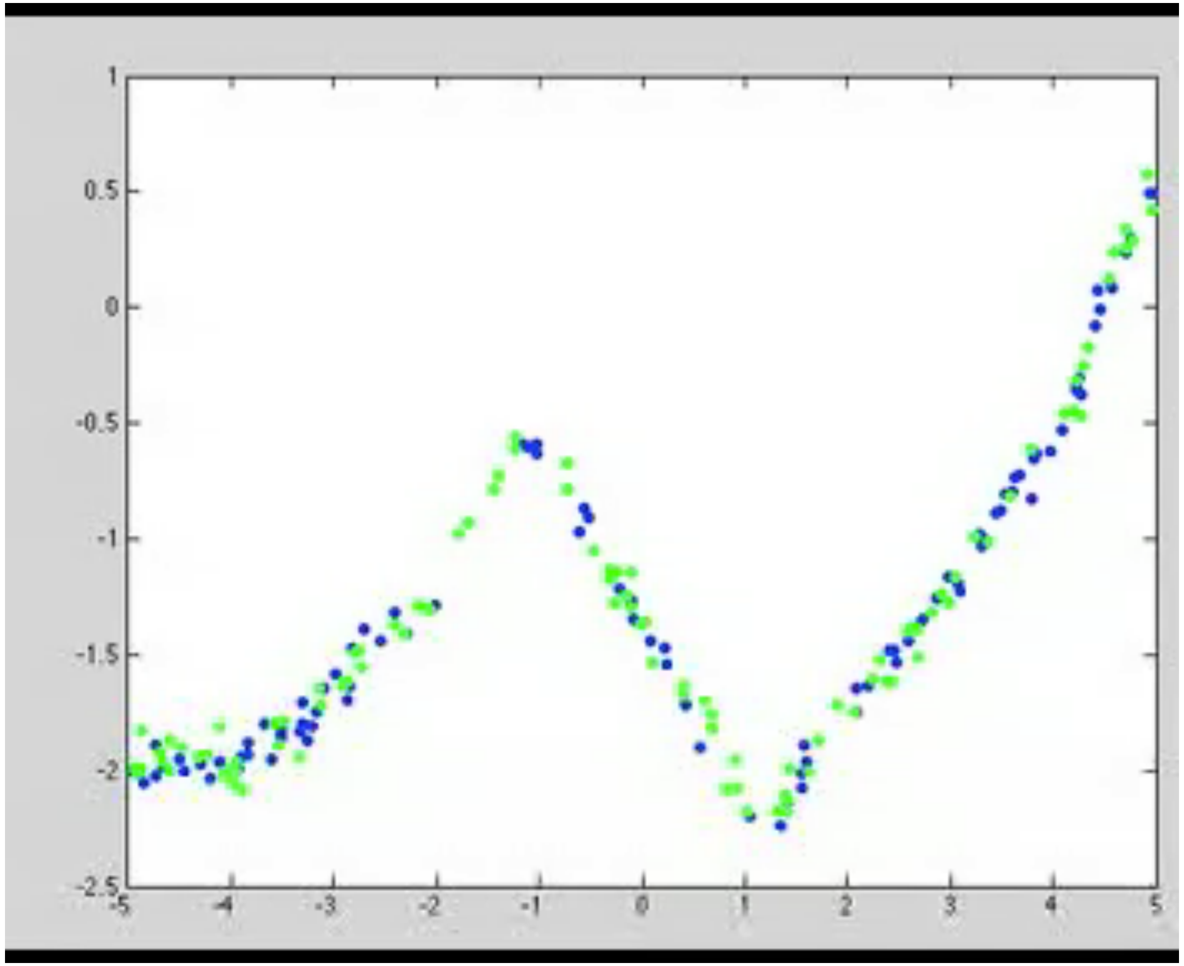
Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



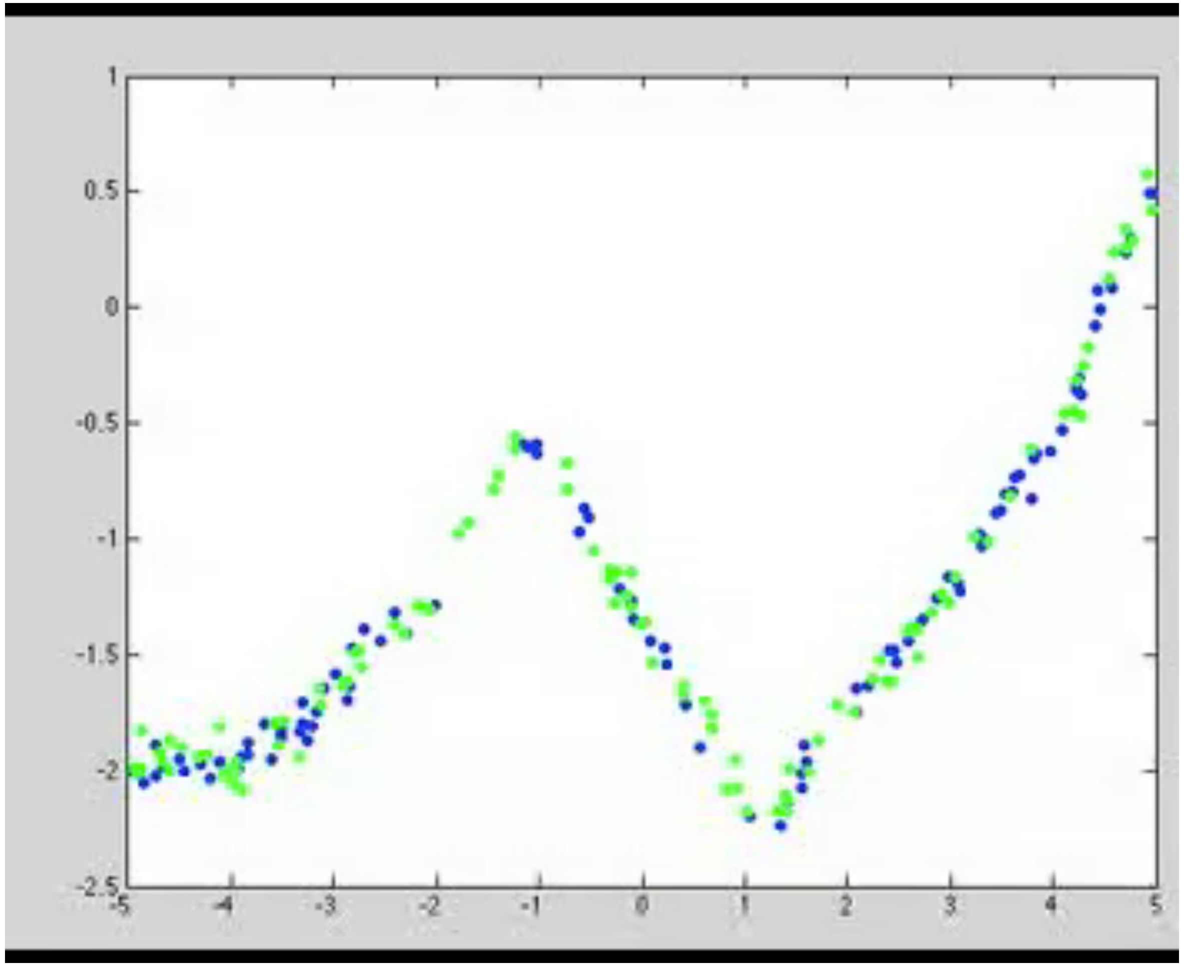
Could add <u>bias</u> and linear basis:

$$Z = \begin{bmatrix} 1 & - x_1 - & g(\|x_1 - x_1\|) \cdots g(\|x_1 - x_n\|) \\ 1 & - x_2 - & \vdots \quad \ddots \quad \vdots \\ 1 & - x_3 - & \\ \vdots & \vdots & \\ \underbrace{1}_{1} & \underbrace{- x_n -}_{d} & \underbrace{g(\|x_1 - x_n\|) \cdots g(\|x_n - x_n\|)}_{n} \end{bmatrix}$$

This reverts to <u>linear regression</u> instead of 0 away from data.

Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



Could add bias and linear basis:

$$Z = \begin{bmatrix} 1 & — x_1 — & g(\|x_1 - x_1\|) & \cdots & g(\|x_1 - x_n\|) \\ 1 & — x_2 — & \vdots & \ddots & \vdots \\ 1 & — x_3 — & & & \\ \vdots & \vdots & & & \\ 1 & — x_n — & g(\|x_1 - x_n\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}$$

$$\underbrace{}_{1} \quad \underbrace{}_{d} \quad \underbrace{}_{n}$$

This reverts to linear regression instead of 0 away from data.

RBFs and Regularization

- Gaussian Radial basis functions (RBFs) predictions:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

RBFs and Regularization

- Gaussian Radial basis functions (RBFs) predictions:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

RBFs and Regularization

- Gaussian Radial basis functions (RBFs) predictions:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

  – Flexible bases that can model any continuous function.

    - Quick almost-proof: Z is square, almost always invertible, so least squares gives $Z v = Z (Z^T Z)^{-1} Z^T y = Z Z^{-1} Z^{-T} Z^T y = y$

    bonus!

  – But with 'n' data points RBFs have 'n' basis functions.

RBFs and Regularization

- Gaussian Radial basis functions (RBFs) predictions:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

  – Flexible bases that can model any continuous function.

    - Quick almost-proof: Z is square, almost always invertible, so least squares gives $Z v = Z (Z^T Z)^{-1} Z^T y = Z Z^{-1} Z^{-T} Z^T y = y$ bonus!

  – But with 'n' data points RBFs have 'n' basis functions.

- How do we avoid overfitting with this huge number of features?

RBFs and Regularization

- Gaussian Radial basis functions (RBFs) predictions:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

  – Flexible bases that can model any continuous function.

    - Quick almost-proof: Z is square, almost always invertible, so least squares gives $Z v = Z (Z^T Z)^{-1} Z^T y = Z Z^{-1} Z^{-T} Z^T y = y$ bonus!

  – But with ‘n’ data points RBFs have ‘n’ basis functions.

- How do we avoid overfitting with this huge number of features?
  – We regularize ‘w’ and use validation error to choose σ and λ.

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

for each value of λ and σ:

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

for each value of λ and σ:

- Compute Z on training data (and σ)

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

for each value of λ and σ:

- Compute Z on training data (and σ)
- Compute best v: $v = (Z^T Z + \lambda I)^{-1} Z^T y$

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

for each value of λ and σ:

- Compute Z on training data (and σ)
- Compute best v: $v = (Z^T Z + \lambda I)^{-1} Z^T y$
- Compute $\tilde{Z}$ on validation data (using train data distances)

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

for each value of λ and σ:

- Compute Z on training data (and σ)
- Compute best v: $v = (Z^T Z + \lambda I)^{-1} Z^T y$
- Compute $\tilde{Z}$ on validation data (using train data distances)
- Make predictions $\hat{y} = \underset{t \times n}{\tilde{Z}} \underset{n \times 1}{v}$

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

for each value of λ and σ:

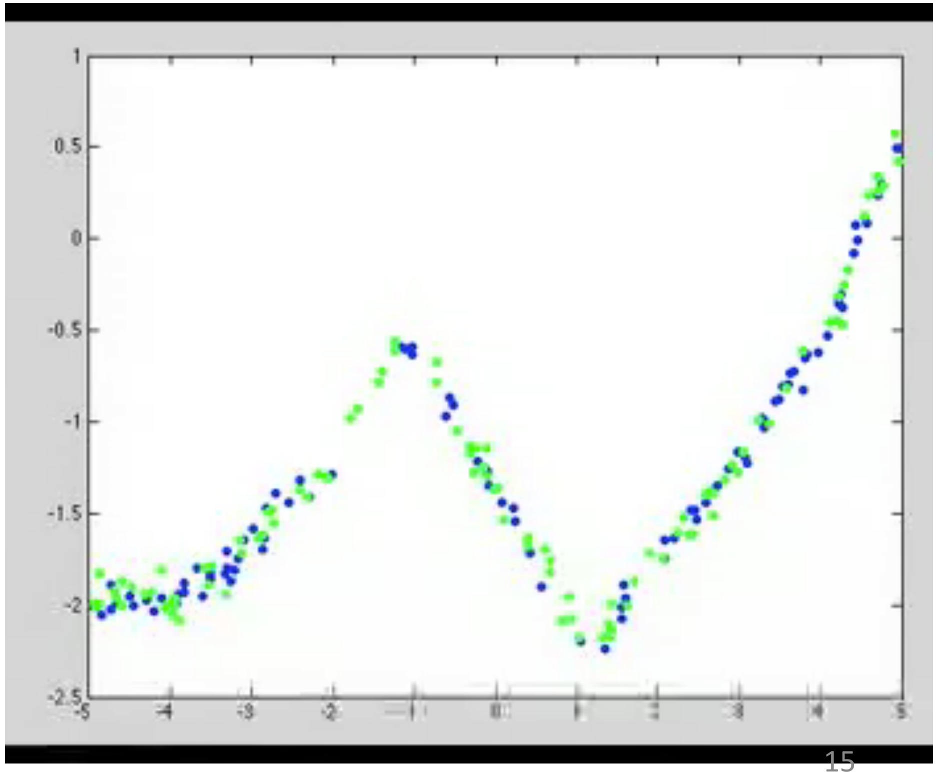
- Compute Z on training data (and σ)
- Compute best v: $v = (Z^T Z + \lambda I)^{-1} Z^T y$
- Compute $\tilde{Z}$ on validation data (using train data distances)
- Make predictions $\hat{y} = \underbrace{\tilde{Z}}_{t \times n} \underbrace{v}_{n \times 1}$
- Compute validation error $\|\hat{y} - \tilde{y}\|^2$

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

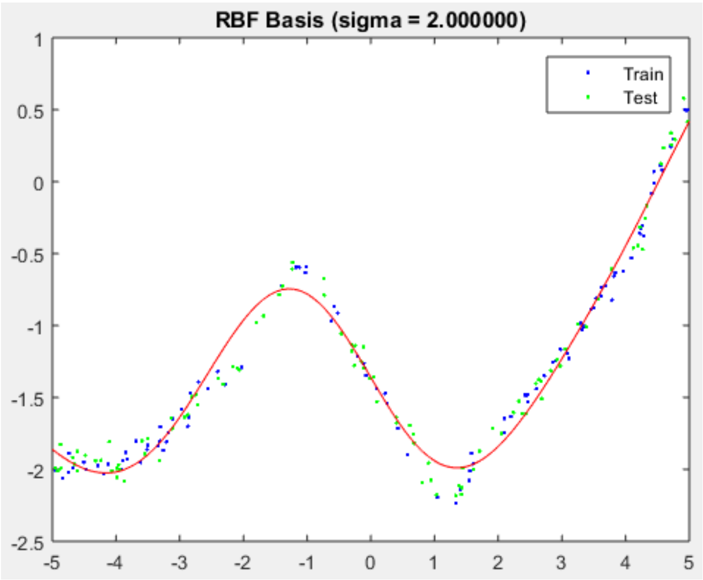
for each value of λ and σ:

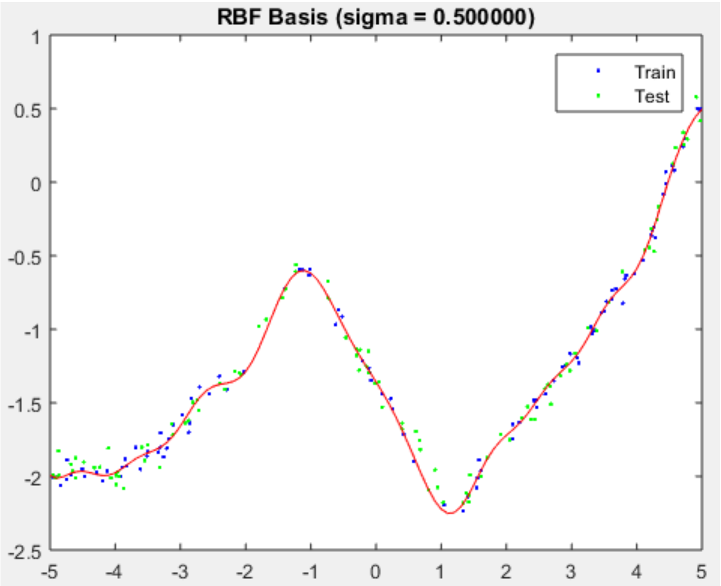
- Compute Z on training data (and σ)
- Compute best v: $v = (Z^T Z + \lambda I)^{-1} Z^T y$
- Compute $\tilde{Z}$ on validation data (using train data distances)
- Make predictions $\hat{y} = \underbrace{\tilde{Z}}_{t \times n} \underbrace{v}_{n \times 1}$
- Compute validation error $\|\hat{y} - \tilde{y}\|^2$

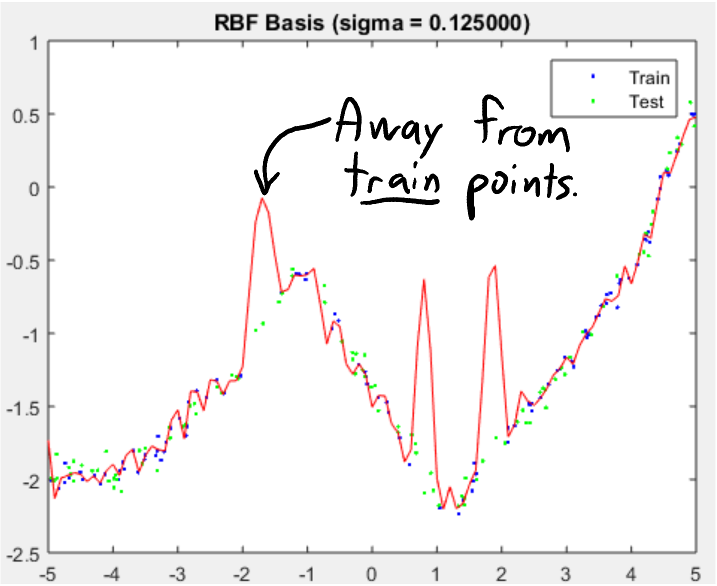


RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!







  - Expensive at test time: needs distance to all training examples.

Hyper-Parameter Optimization

- In this setting we have 2 hyper-parameters (σ and λ).

Hyper-Parameter Optimization

- In this setting we have 2 hyper-parameters (σ and λ).
- More complicated models have even more hyper-parameters.
  - This makes searching all values expensive (increases over-fitting risk).

- Leads to the problem of hyper-parameter optimization.
  - Try to efficiently find “best” hyper-parameters.

Hyper-Parameter Optimization

- In this setting we have 2 hyper-parameters (σ and λ).
- More complicated models have even more hyper-parameters.
  - This makes searching all values expensive (increases over-fitting risk).

- Leads to the problem of hyper-parameter optimization.
  - Try to efficiently find “best” hyper-parameters.

- Simplest approaches:
  - Exhaustive search: try all combinations among a fixed set of σ and λ values.
  - Random search: try random values.

bonus!

Hyper-Parameter Optimization

- Other common hyper-parameter optimization methods:
  - Exhaustive search with pruning:
    - If it “looks” like test error is getting worse as you decrease λ, stop decreasing it.

bonus!

Hyper-Parameter Optimization

- Other common hyper-parameter optimization methods:
  - Exhaustive search with pruning:
    - If it "looks" like test error is getting worse as you decrease λ, stop decreasing it.
  - Coordinate search:
    - Optimize one hyper-parameter at a time, keeping the others fixed.
    - Repeatedly go through the hyper-parameters
  - Stochastic local search:
    - Generic global optimization methods (simulated annealing, genetic algorithms, etc.).

bonus!

Hyper-Parameter Optimization

- Other common hyper-parameter optimization methods:
  - Exhaustive search with pruning:
    - If it "looks" like test error is getting worse as you decrease λ, stop decreasing it.
  - Coordinate search:
    - Optimize one hyper-parameter at a time, keeping the others fixed.
    - Repeatedly go through the hyper-parameters
  - Stochastic local search:
    - Generic global optimization methods (simulated annealing, genetic algorithms, etc.).
  - Bayesian optimization (Mike's PhD research topic):
    - Use (e.g.) RBF regression to build model of how hyper-parameters affect validation error.
    - Try the best guess based on the model.

(pause)

Previously: Search and Score

- We talked about search and score for feature selection:
  - Define a “score” and “search” for features with the best score.

Previously: Search and Score

- We talked about search and score for feature selection:
  - Define a "score" and "search" for features with the best score.
- Usual scores count the number of non-zeroes ("L0-norm"):

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda \underbrace{\|w\|_0}_{\text{number of non-zeroes in 'w'}}$$

Previously: Search and Score

- We talked about search and score for feature selection:
  - Define a "score" and "search" for features with the best score.
- Usual scores count the number of non-zeroes ("L0-norm"):

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda \underbrace{\|w\|_0}_{\text{number of non-zeroes in 'w'}}$$

- But it's hard to find the 'w' minimizing this objective.
- We discussed forward selection, but requires fitting $O(d^2)$ models.

Previously: Search and Score

- What if we want to pick among millions or billions of variables?

Previously: Search and Score

- What if we want to pick among millions or billions of variables?

- If ‘d’ is large, forward selection is too slow:
  - For least squares, need to fit $O(d^2)$ models. Imagine d = 10^6.

Previously: Search and Score

- What if we want to pick among millions or billions of variables?

- If ‘d’ is large, forward selection is too slow:
  - For least squares, need to fit $O(d^2)$ models. Imagine d = 10^6.

- The situation is worse if we aren’t using basic least squares:
  - With regularization, for every lambda, we need to fit $O(d^2)$ models.

L1-Regularization

- Instead of L0- or L2-norm, consider regularizing by the L1-norm:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

L1-Regularization

- Instead of L0- or L2-norm, consider regularizing by the L1-norm:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

- Like L2-norm, it's convex and improves our test error.

L1-Regularization

- Instead of L0- or L2-norm, consider regularizing by the L1-norm:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

- Like L2-norm, it's convex and improves our test error.
- Like L0-norm, it encourages elements of 'w' to be exactly zero.

L1-Regularization

- Instead of L0- or L2-norm, consider regularizing by the L1-norm:

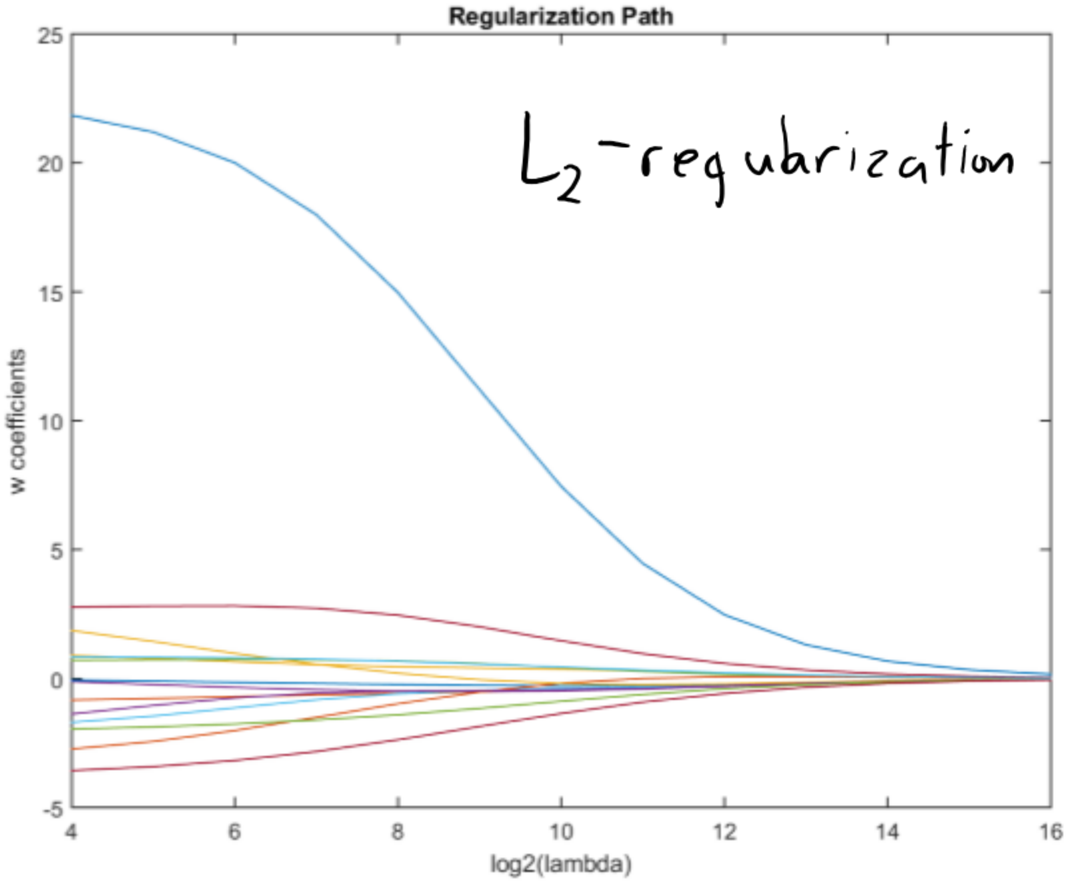
$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

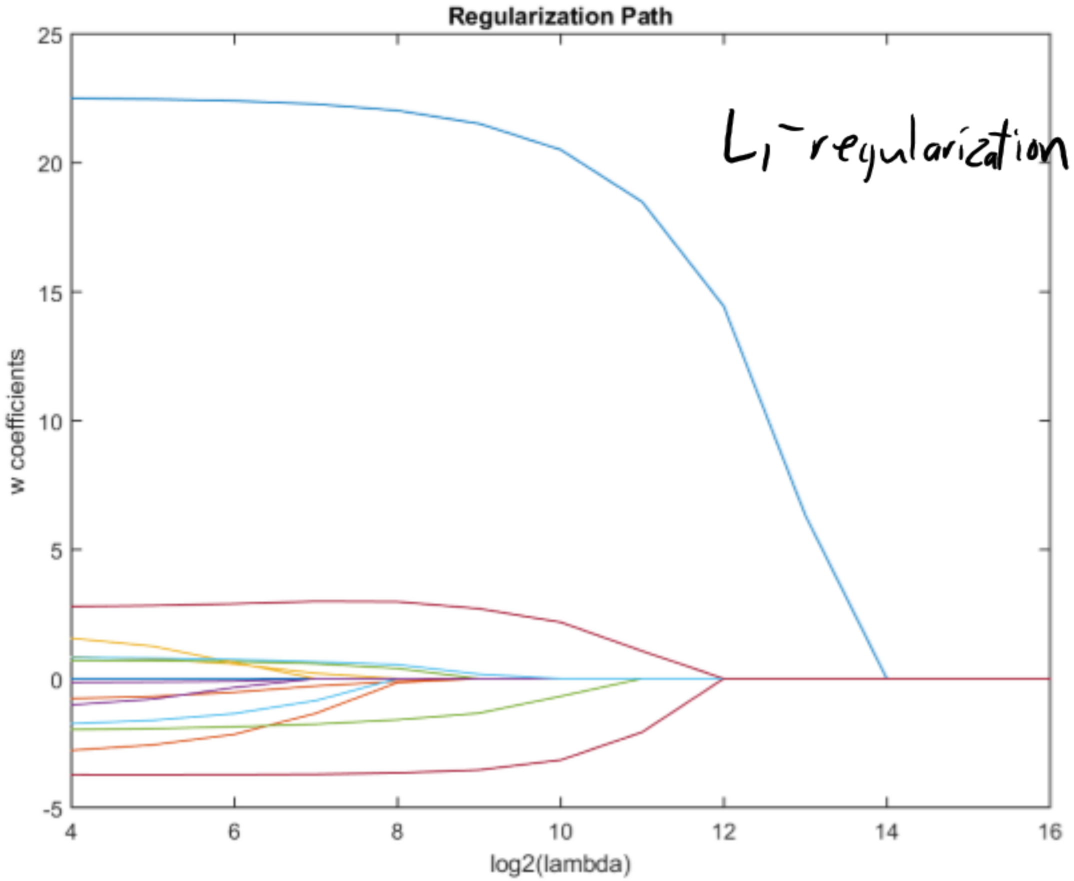
- Like L2-norm, it's convex and improves our test error.
- Like L0-norm, it encourages elements of 'w' to be exactly zero.

- L1-regularization simultaneously regularizes and selects features.
  - **Very fast alternative to search and score.**
  - Sometimes called "LASSO" regularization.

L2-Regularization vs. L1-Regularization

- Regularization path of w_j values as 'λ' varies:





- L1-Regularization sets values to exactly 0 (next slides explore why).

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn’t.
  - But don’t they both shrink variables towards zero?

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables towards zero?

- What is the penalty for setting $w_j = 0.00001$?

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables towards zero?
- What is the penalty for setting $w_j = 0.00001$?
- L0-regularization: penalty of λ.
  - A constant penalty for any non-zero value.
  - Encourages you to set w_j exactly to zero, but otherwise doesn't care if w_j is small or not.

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables towards zero?

- What is the penalty for setting $w_j = 0.00001$?

- L0-regularization: penalty of λ.
  - A constant penalty for any non-zero value.
  - Encourages you to set w_j exactly to zero, but otherwise doesn't care if w_j is small or not.

- L2-regularization: penalty of $(\lambda/2)(0.00001) = 0.0000000005\lambda$.
  - The penalty gets smaller as you get closer to zero.
  - The penalty asymptotically vanishes as w_j approaches 0 (no incentive for "exact" zeroes).

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables towards zero?

- What is the penalty for setting $w_j = 0.00001$?

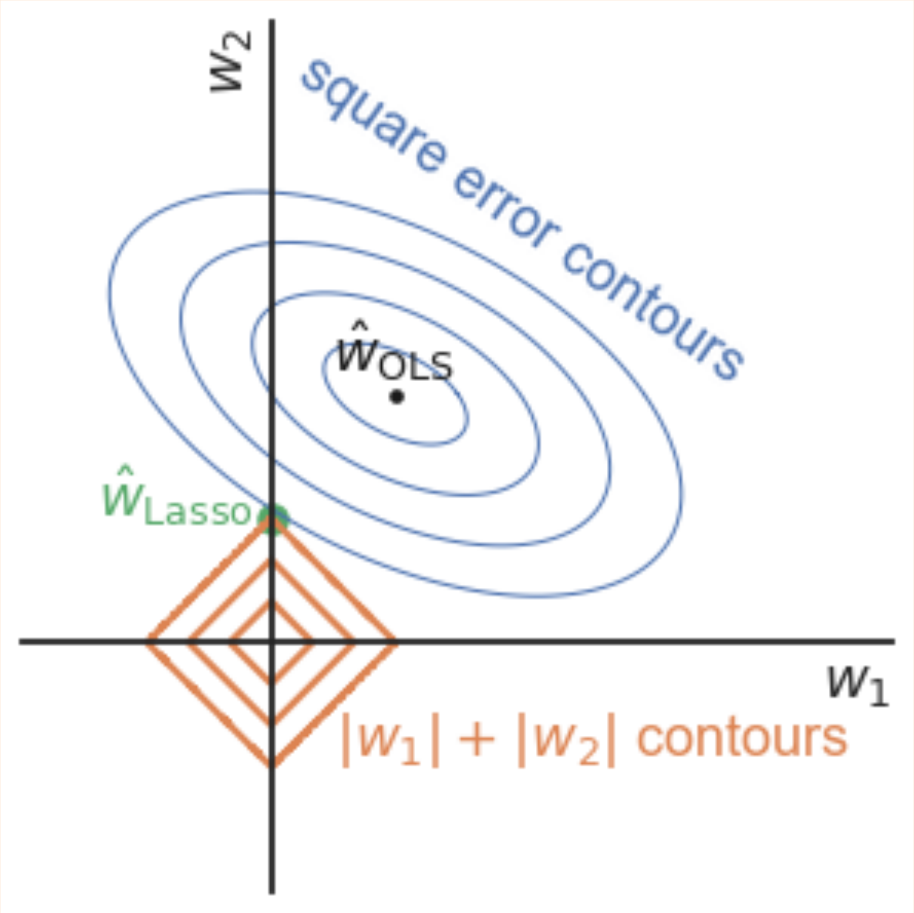
- L0-regularization: penalty of λ.
  - A constant penalty for any non-zero value.
  - Encourages you to set w_j exactly to zero, but otherwise doesn't care if w_j is small or not.

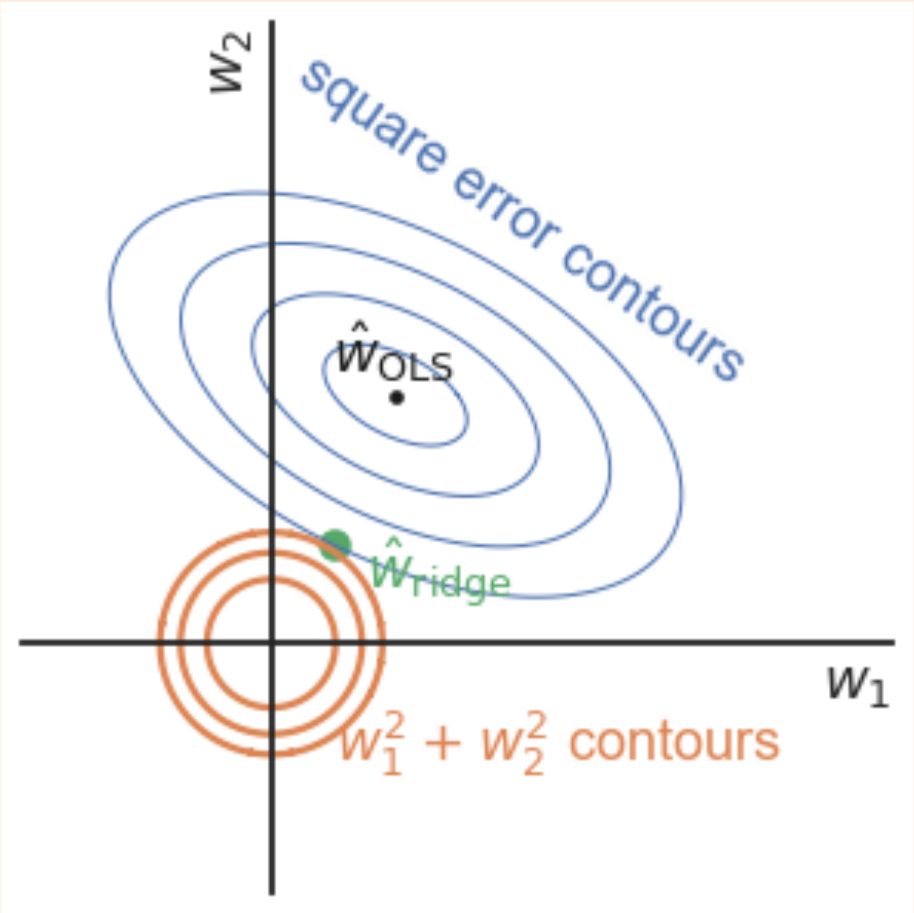
- L2-regularization: penalty of $(\lambda/2)(0.00001) = 0.0000000005\lambda$.
  - The penalty gets smaller as you get closer to zero.
  - The penalty asymptotically vanishes as w_j approaches 0 (no incentive for "exact" zeroes).

- L1-regularization: penalty of $\lambda|0.00001| = 0.00001\lambda$.
  - The penalty stays is proportional to how far away w_j is from zero.
  - There is still something to be gained from making a tiny value exactly equal to 0.

bonus!

Regularizers and Sparsity





Loss plus error usually minimized at "corners" (sparse points)

Minimizer moved towards 0, but axis-independently

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.

- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All 'w_j' tend to be non-zero.
- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.
  - Many 'w_j' tend to be zero.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All 'w_j' tend to be non-zero.
  - Can learn with *linear* number of irrelevant features.
    - E.g., only O(d) relevant features.

- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.
  - Many 'w_j' tend to be zero.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All 'w_j' tend to be non-zero.
  - Can learn with *linear* number of irrelevant features.
    - E.g., only O(d) relevant features.

- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.
  - Many 'w_j' tend to be zero.
  - Can learn with **exponential** number of irrelevant features.
    - E.g., only O(log(d)) relevant features.
      Paper on this result by Andrew Ng

bonus!

L1-Regularization Applications

- Used to give super-resolution in imaging black holes.
  - Sparsity arises in a particular basis.

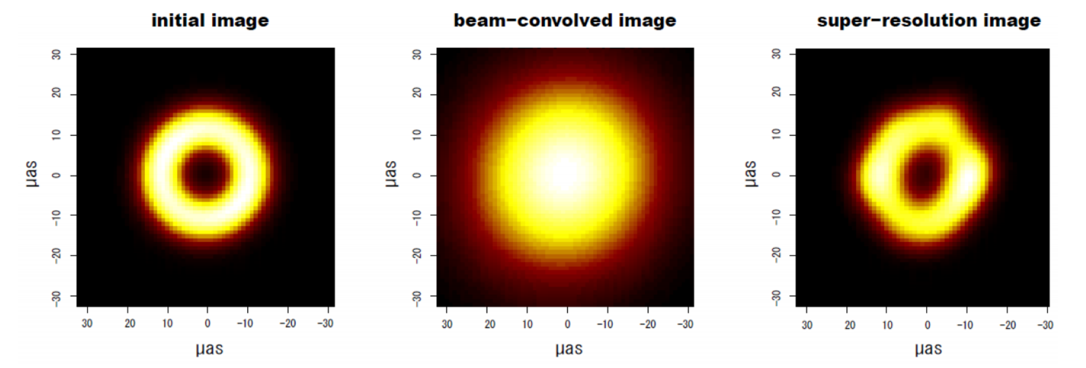


Figure 2. Simulated images of M87. From left to right, the initial model, the image with 0-filling, and the image with LASSO. Improvement of resolution in the LASSO image is significant.

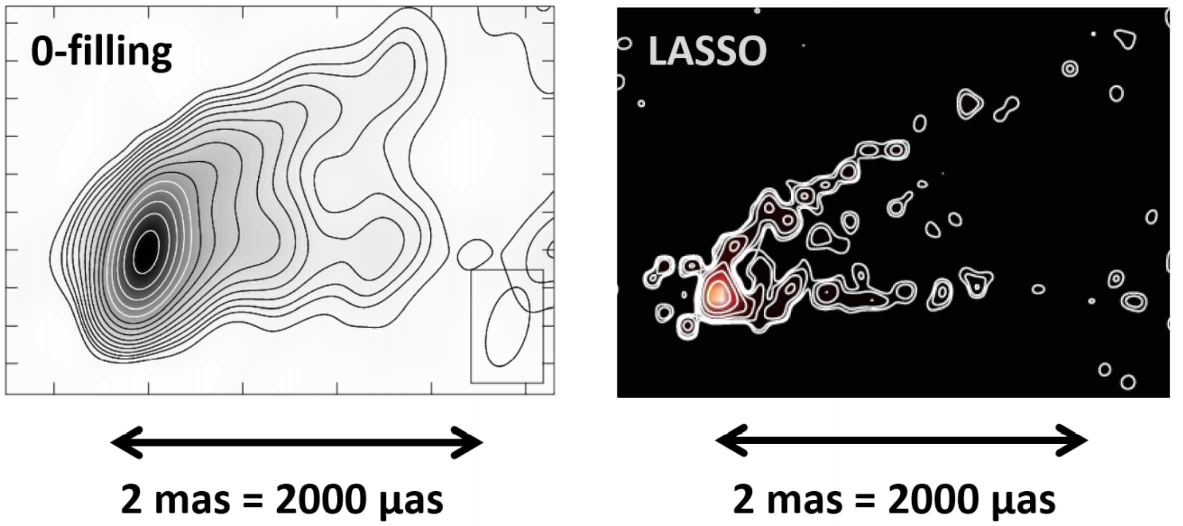


Figure 3. Standard and LASSO images of M87 observed with VLBA at a wavelength of 7 mm. In the two plots, exactly the same data are used. The angular resolution is better in the LASSO image, and the detailed structure of the M87 jet can be traced in more detail.

- Another application:
  - Use L1-regularization with Gaussian RBFs to reduce prediction time.

https://iopscience.iop.org/article/10.1088/1742-6596/699/1/012006/pdf

L1-loss vs. L1-regularization

- Don't confuse the L1 loss with L1-regularization!
  - L1-loss is robust to outlier data points.
    - You can use this instead of removing outliers.
  - L1-regularization is robust to irrelevant features.
    - You can use this instead of removing features.

L1-loss vs. L1-regularization

- Don't confuse the L1 loss with L1-regularization!
  - L1-loss is robust to outlier data points.
    - You can use this instead of removing outliers.
  - L1-regularization is robust to irrelevant features.
    - You can use this instead of removing features.
- And note that you can be robust to outliers and irrelevant features:

$$f(w) = \underbrace{\|Xw - y\|_1}_{L_1\text{-loss}} + \lambda \underbrace{\|w\|_1}_{L_1\text{-regularizer}}$$

L1-loss vs. L1-regularization

- Don't confuse the L1 loss with L1-regularization!
  - L1-loss is robust to outlier data points.
    - You can use this instead of removing outliers.
  - L1-regularization is robust to irrelevant features.
    - You can use this instead of removing features.
- And note that you can be robust to outliers and irrelevant features:

$$f(w) = \underbrace{\|Xw - y\|_1}_{L_1\text{-loss}} + \underbrace{\lambda \|w\|_1}_{L_1\text{-regularizer}}$$

- Can we smooth and use "Huber regularization"?
  - Huber regularizer is still robust to irrelevant features.
  - But it's the non-smoothness that sets weights to exactly 0.

L*-Regularization

- L0-regularization (AIC, BIC, Mallow's Cp, Adjusted R^2, ANOVA):
  - Adds penalty on the number of non-zeros to select features.

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_0$$

L*-Regularization

- L0-regularization (AIC, BIC, Mallow's Cp, Adjusted R^2, ANOVA):
  - Adds penalty on the number of non-zeros to select features.

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_0$$

- L2-regularization (ridge regression):
  - Adding penalty on the L2-norm of 'w' to decrease overfitting:

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

L*-Regularization

- L0-regularization (AIC, BIC, Mallow's Cp, Adjusted R^2, ANOVA):
  - Adds penalty on the number of non-zeros to select features.

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_0$$

- L2-regularization (ridge regression):
  - Adding penalty on the L2-norm of 'w' to decrease overfitting:

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

- L1-regularization (LASSO):
  - Adding penalty on the L1-norm decreases overfitting and selects features:

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_1$$

L0- vs. L1- vs. L2-Regularization

	Sparse 'w' (Selects Features)	Speed	Unique 'w'	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

- L1-Regularization isn't as sparse as L0-regularization.

L0- vs. L1- vs. L2-Regularization

	Sparse 'w' (Selects Features)	Speed	Unique 'w'	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

- L1-Regularization isn't as sparse as L0-regularization.
  - L1-regularization tends to give more false positives (selects too many).

L0- vs. L1- vs. L2-Regularization

	Sparse 'w' (Selects Features)	Speed	Unique 'w'	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

- L1-Regularization isn't as sparse as L0-regularization.
  - L1-regularization tends to give more false positives (selects too many).
  - And it's only "fast" and "1 line" with specialized solvers.

L0- vs. L1- vs. L2-Regularization

	Sparse 'w' (Selects Features)	Speed	Unique 'w'	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

- L1-Regularization isn't as sparse as L0-regularization.
  – L1-regularization tends to give more false positives (selects too many).
  – And it's only "fast" and "1 line" with specialized solvers.
- "Elastic net" (L1- and L2-regularization) is sparse, fast, and unique. [https://www.stat.cmu.edu/~ryantibs/papers/lassounique.pdf]

L0- vs. L1- vs. L2-Regularization

	Sparse ‘w’ (Selects Features)	Speed	Unique ‘w’	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

- L1-Regularization isn’t as sparse as L0-regularization.
  - L1-regularization tends to give more false positives (selects too many).
  - And it’s only “fast” and “1 line” with specialized solvers.
- “Elastic net” (L1- and L2-regularization) is sparse, fast, and unique. [https://www.stat.cmu.edu/~ryantibs/papers/lassounique.pdf]
- Using L0+L2 does not give a unique solution.

Summary

Summary

- Radial basis functions:
  - Non-parametric bases that can model any function.

Summary

- Radial basis functions:
  - Non-parametric bases that can model any function.
- L1-regularization:
  - Simultaneous regularization and feature selection.
  - Robust to having lots of irrelevant features.

Summary

- Radial basis functions:
  - Non-parametric bases that can model any function.
- L1-regularization:
  - Simultaneous regularization and feature selection.
  - Robust to having lots of irrelevant features.
- Next time: are we really going to use regression for classification?

bonus!

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?

bonus!

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

bonus!

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- Without regularization, we could choose any of these 3.
  - They all have same error, so regularization will "break tie".

bonus!

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- Without regularization, we could choose any of these 3.
  - They all have same error, so regularization will "break tie".
- With L0-regularization, we would choose w^2:

Regularizers and Sparsity

bonus!

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- Without regularization, we could choose any of these 3.
  - They all have same error, so regularization will "break tie".
- With L0-regularization, we would choose w^2:

$$\|w^1\|_0 = 2 \qquad \|w^2\|_0 = 1 \qquad \|w^3\|_0 = 2$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

$$\|w^1\|^2 = 100^2 + 0.02^2$$
$$= 10000.0004$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.62 \end{bmatrix}$$

$$\|w^1\|^2 = 100^2 + 0.02^2 = 10000.0004$$

$$\|w^2\|^2 = 100^2 + 0^2 = 10000$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

$$\|w^1\|^2 = 100^2 + 0.02^2 = 10000.0004$$

$$\|w^2\|^2 = 100^2 + 0^2 = 10000$$

$$\|w^3\|^2 = 99.99^2 + 0.02^2 = 9998.0005$$

bonus!

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- With L2-regularization, we would choose w^3:

$$\|w^1\|^2 = 100^2 + 0.02^2 = 10000.0004 \qquad \|w^2\|^2 = 100^2 + 0^2 = 10000 \qquad \|w^3\|^2 = 99.99^2 + 0.02^2 = 9998.0005$$

- L2-regularization focuses on decreasing largest (makes w_j similar).

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.62 \end{bmatrix}$$

$$\|w^1\|_1 = 100 + 0.02$$
$$= 100.02$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

$$\|w^1\|_1 = 100 + 0.02 = 100.02 \qquad \|w^2\|_1 = 100 + 0 = 100$$

bonus!

Regularizers and Sparsity

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

$$\|w^1\|_1 = 100 + 0.02 = 100.02$$

$$\|w^2\|_1 = 100 + 0 = 100$$

$$\|w^3\|_1 = 99.99 + 0.02 = 100.01$$

bonus!

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink variables to zero?
- Consider problem where 3 vectors can get minimum training error:

$$w^1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \qquad w^2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \qquad w^3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

- With L1-regularization, we would choose w^2:

$$\|w^1\|_1 = 100 + 0.02 = 100.02 \qquad \|w^2\|_1 = 100 + 0 = 100 \qquad \|w^3\|_1 = 99.99 + 0.02 = 100.01$$

- L1-regularization focuses on decreasing all w_j until they are 0.

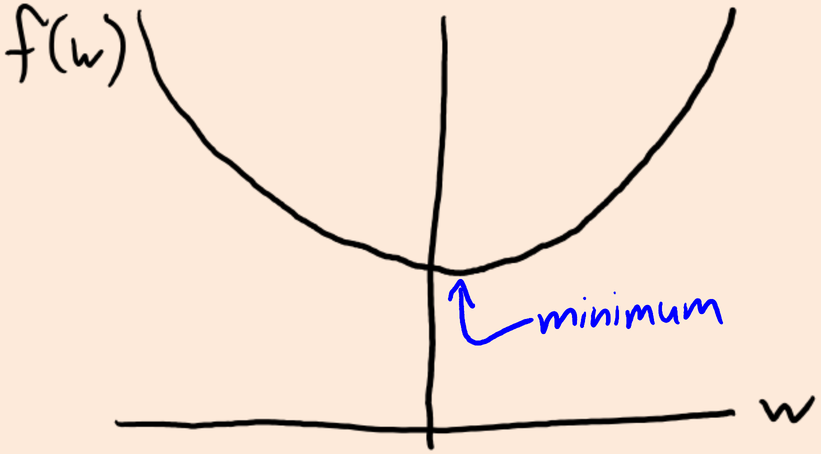
bonus!

Sparsity and Least Squares

- Consider 1D least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2$$

- This is a convex 1D quadratic function of ‘w’ (i.e., a parabola):



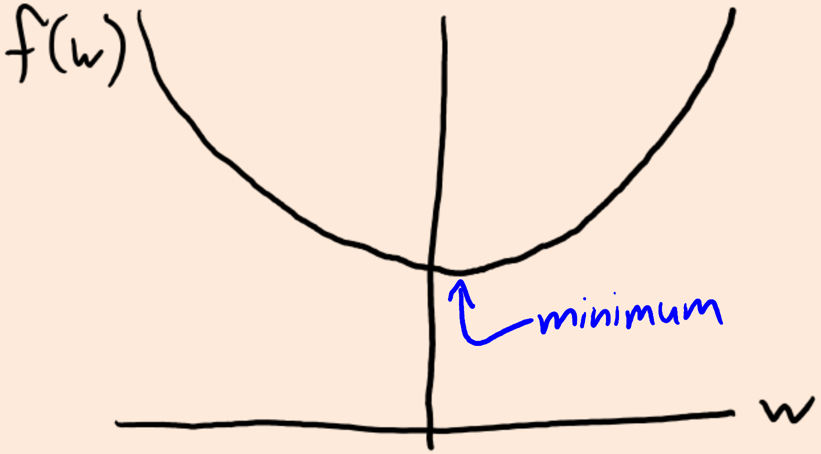
Sparsity and Least Squares

bonus!

- Consider 1D least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2$$

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



- This variable does not look relevant (minimum is close to 0).
  - But for finite 'n' the minimum is unlikely to be exactly zero.

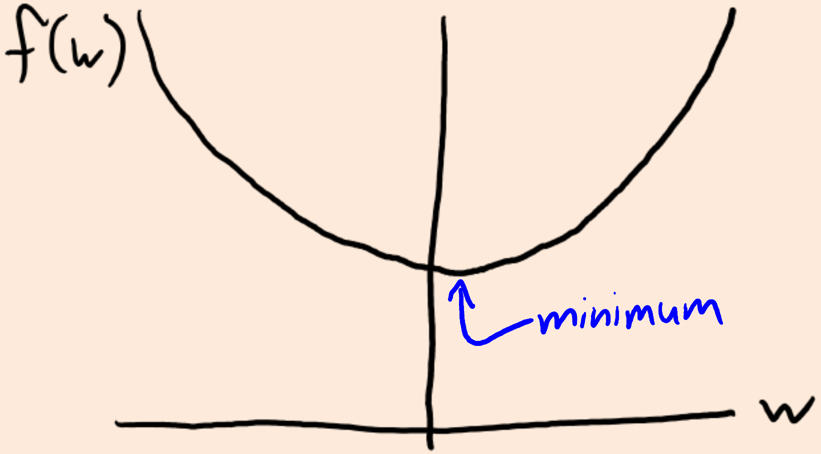
bonus!

Sparsity and Least Squares

- Consider 1D least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2$$

- This is a convex 1D quadratic function of ‘w’ (i.e., a parabola):



$f'(0) = 0$ only happens if $\sum_{i=1}^{n} y_i x_i = 0$. (bonus)

- This variable does not look relevant (minimum is close to 0).
  – But for finite ‘n’ the minimum is unlikely to be exactly zero.

bonus!

Sparsity and L0-Regularization

- Consider 1D L0-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \underbrace{\lambda \|w\|_0}_{\begin{cases}\lambda & \text{if } w \neq 0 \\ 0 & \text{if } w = 0\end{cases}}$$

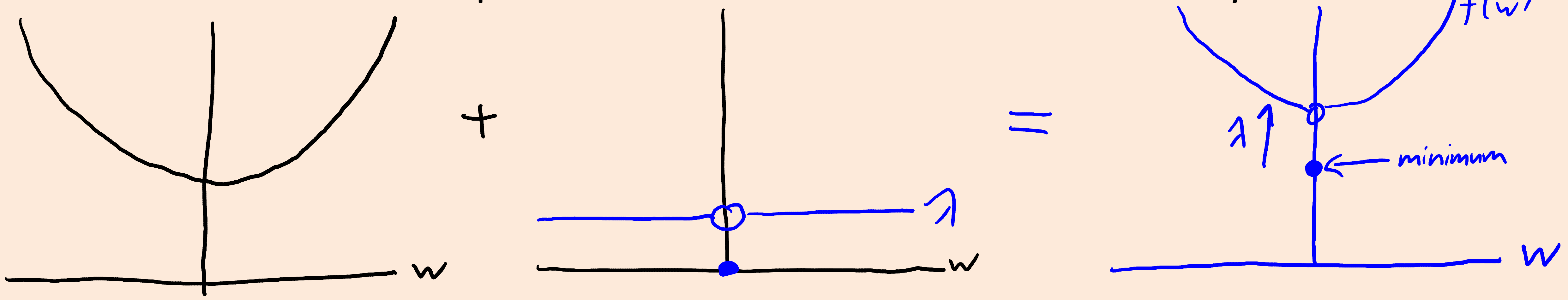
bonus!

Sparsity and L0-Regularization

- Consider 1D L0-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \underbrace{\lambda \|w\|_0}_{\substack{\lambda \text{ if } w \neq 0 \\ 0 \text{ if } w = 0}}$$

- This is a convex 1D quadratic function but with a discontinuity at 0:



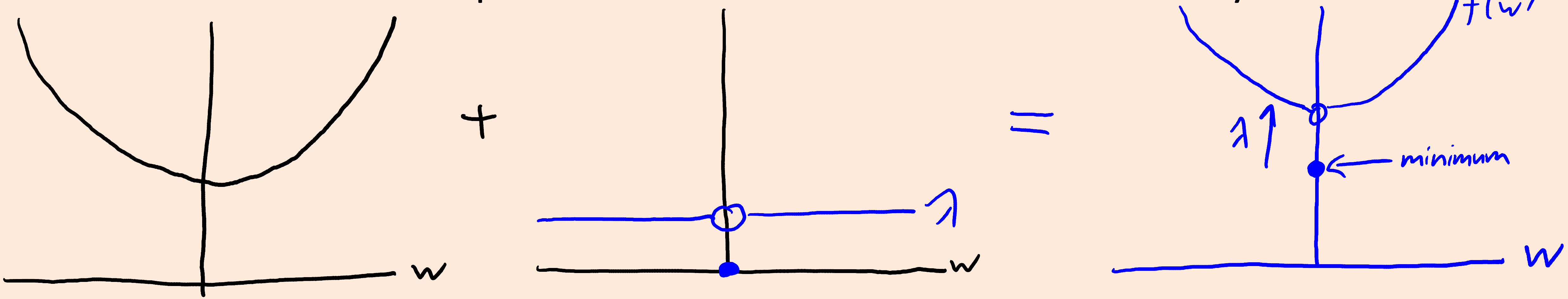
Sparsity and L0-Regularization

bonus!

- Consider 1D L0-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \underbrace{\lambda \|w\|_0}_{\begin{cases}\lambda & \text{if } w \neq 0\\ 0 & \text{if } w = 0\end{cases}}$$

- This is a convex 1D quadratic function but with a discontinuity at 0:



- L0-regularized minimum is often exactly at the 'discontinuity' at 0:
  - Sets the feature to exactly 0 (does feature selection), but is non-convex.

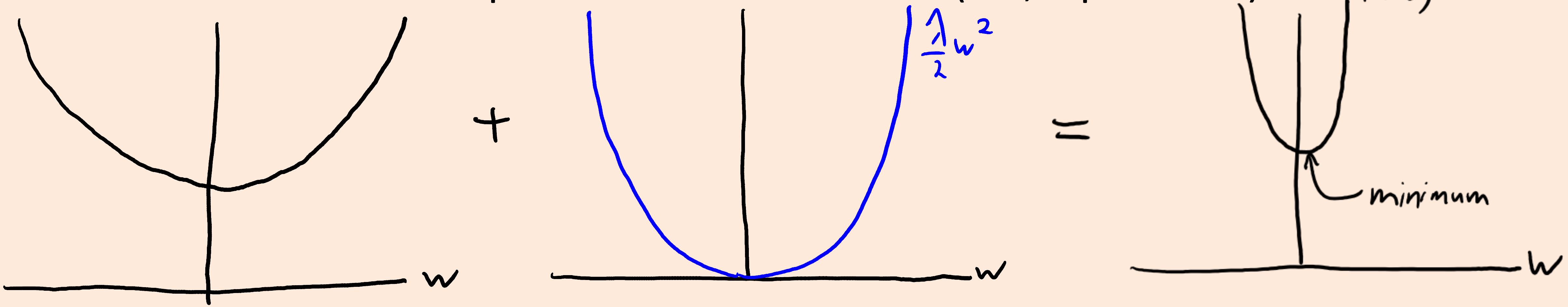
bonus!

Sparsity and L2-Regularization

- Consider 1D L2-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- This is a convex 1D quadratic function of ‘w’ (i.e., a parabola):



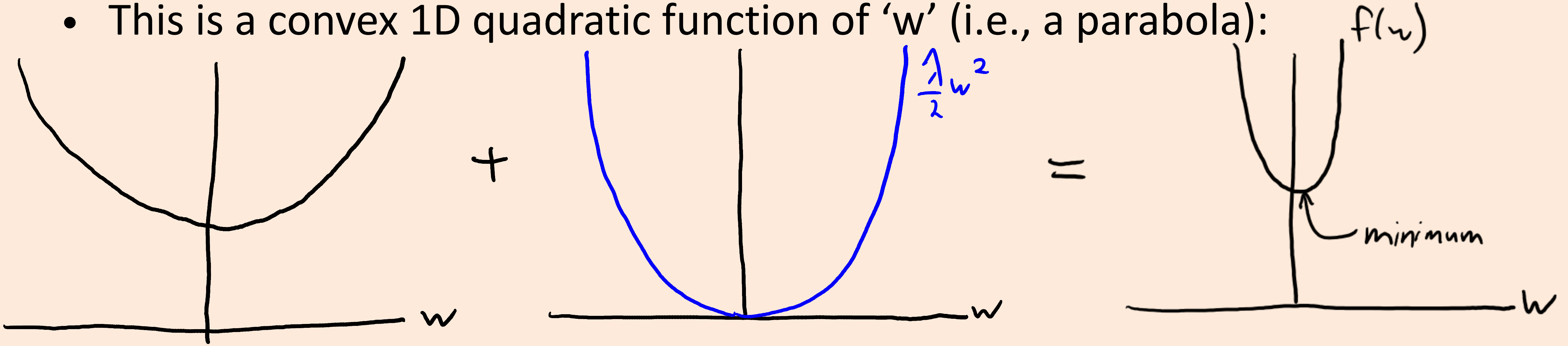
Sparsity and L2-Regularization

bonus!

- Consider 1D L2-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



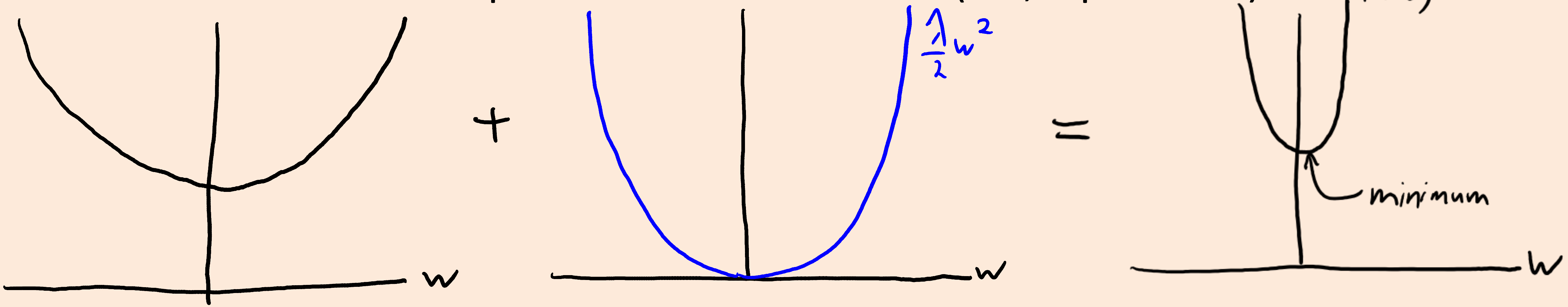
- L2-regularization moves it closer to zero, but not all the way to zero.
  - It doesn't do feature selection ("penalty goes to 0 as slope goes to 0").

Sparsity and L2-Regularization

- Consider 1D L2-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



- L2-regularization moves it closer to zero, but not all the way to zero.
  – It doesn't do feature selection ("penalty goes to 0 as slope goes to 0").

$f'(0) = 0$ only if $\sum_{i=1}^{n} y_i x_i = 0$

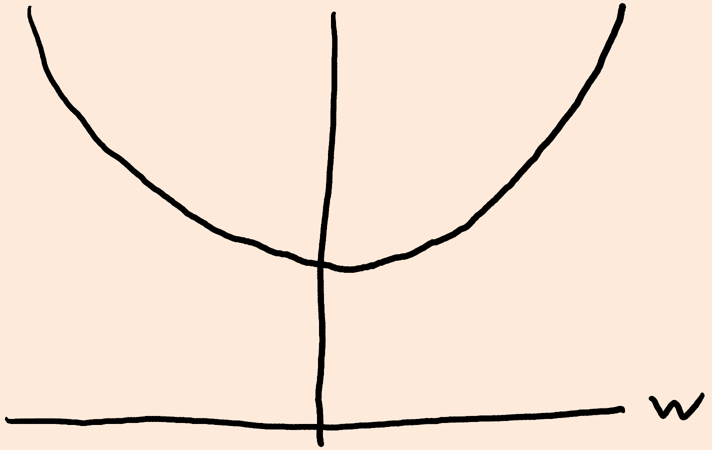
bonus!

Sparsity and L1-Regularization

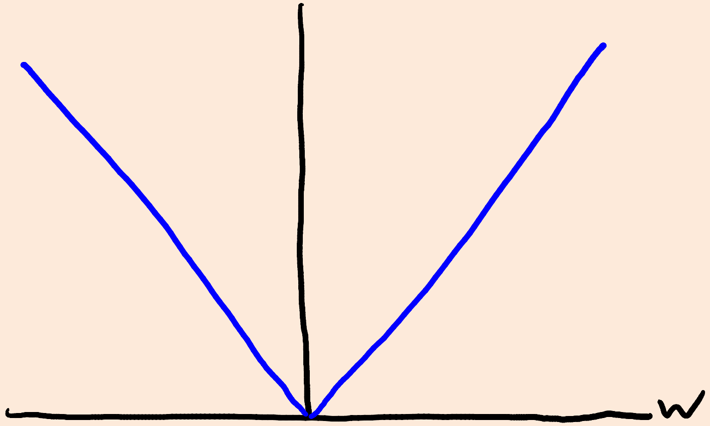
- Consider 1D L1-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \lambda|w|$$

- This is a convex piecwise-quadratic function of ‘w’ with ‘kink’ at 0:



+



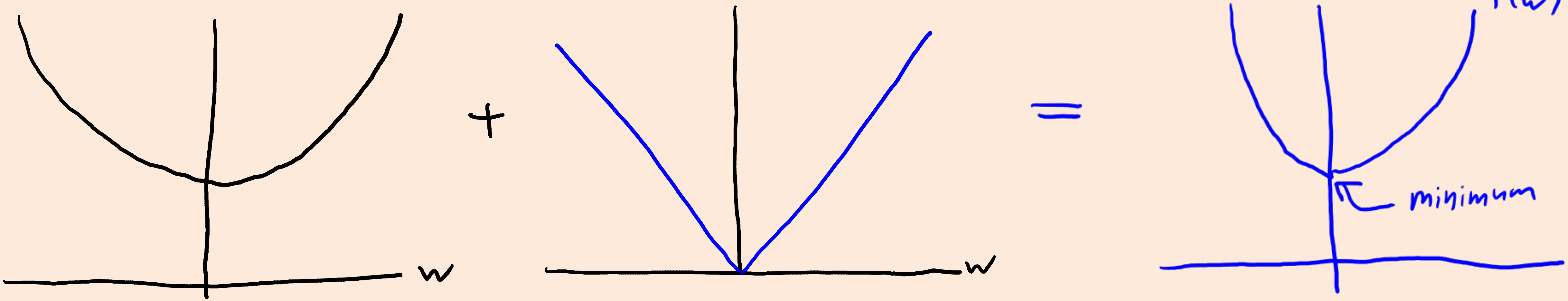
bonus!

Sparsity and L1-Regularization

- Consider 1D L1-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \lambda|w|$$

- This is a convex piecwise-quadratic function of 'w' with 'kink' at 0:



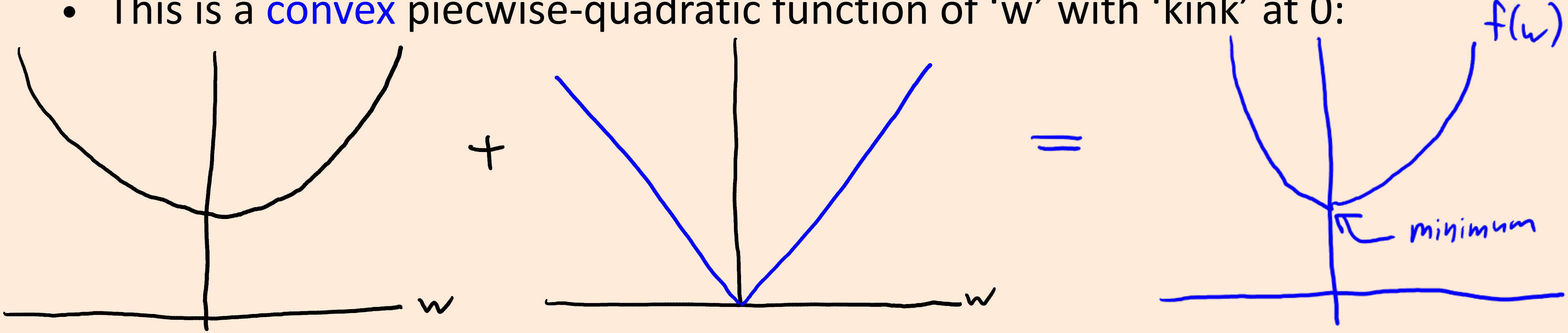
bonus!

Sparsity and L1-Regularization

- Consider 1D L1-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \lambda|w|$$

- This is a convex piecwise-quadratic function of 'w' with 'kink' at 0:



- L1-regularization tends to set variables to exactly 0 (feature selection).
  - Penalty on slope is λ even if you are close to zero.
  - Big λ selects few features, small λ allows many features.

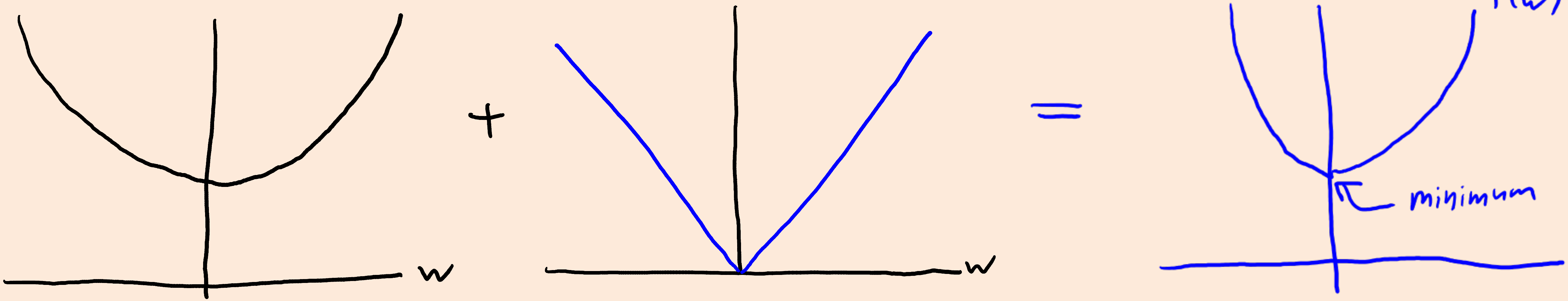
bonus!

Sparsity and L1-Regularization

- Consider 1D L1-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \lambda|w|$$

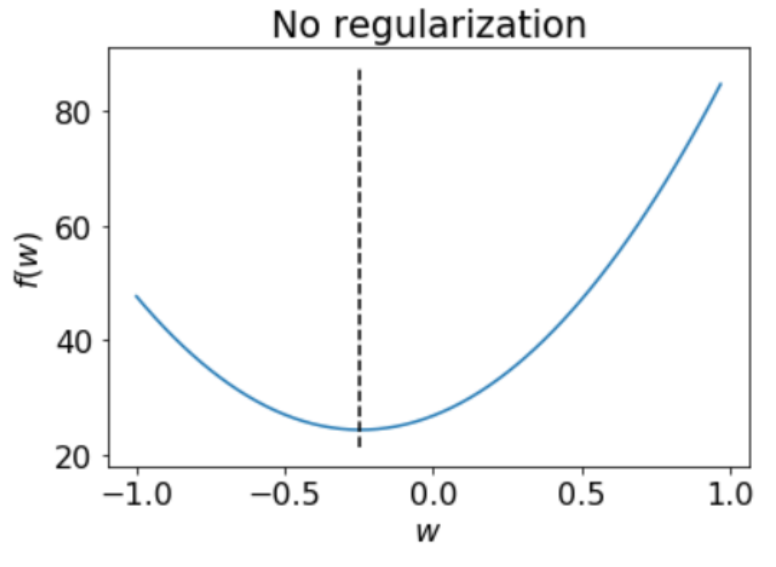
- This is a convex piecwise-quadratic function of 'w' with 'kink' at 0:

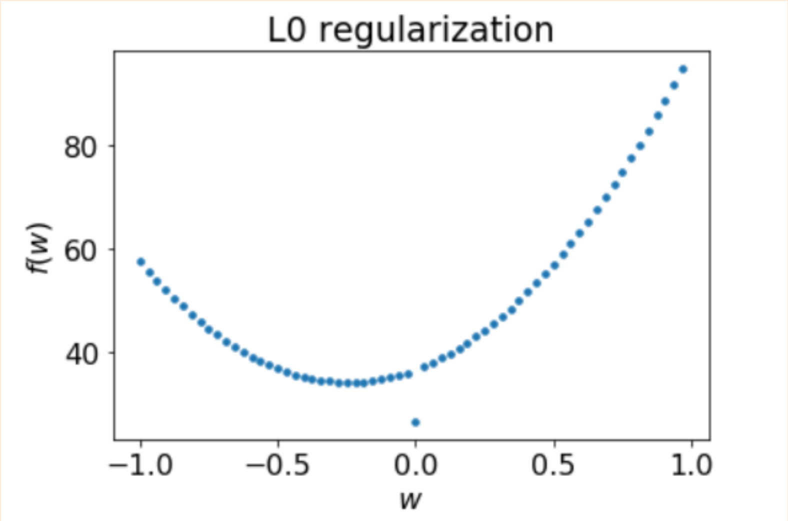


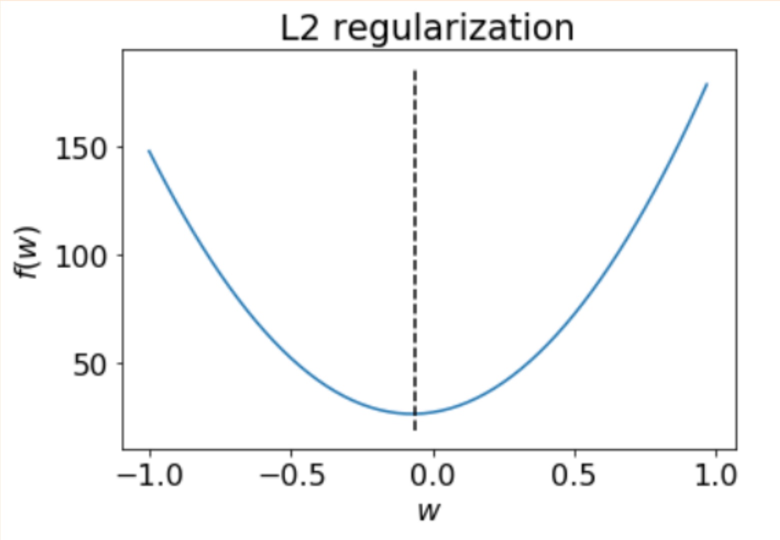
- L1-regularization tends to set variables to exactly 0 (feature selection). Happens when $\left|\sum_{i=1}^{n} x_i y_i\right| \leq \lambda$ (bonus)
  - Penalty on slope is λ even if you are close to zero.
  - Big λ selects few features, small λ allows many features.

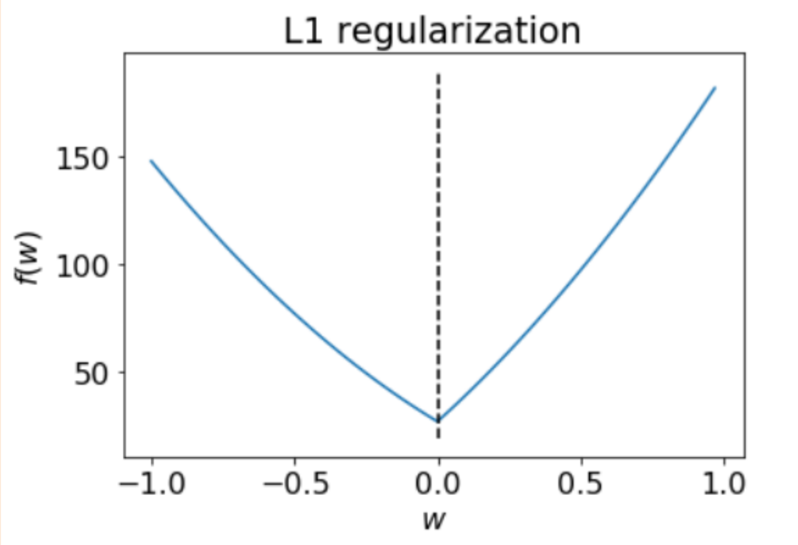
Sparsity and Regularization (with d=1)

bonus!









Why doesn't L2-Regularization set variables to 0?

bonus!

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

bonus!

Why doesn't L2-Regularization set variables to 0?

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

bonus!

Why doesn't L2-Regularization set variables to 0?

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

Set equal to 0: $$\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda w = 0$$

Why doesn't L2-Regularization set variables to 0?

bonus!

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

Set equal to 0: $\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda w = 0$ $\xrightarrow{\text{re-arrange}}$ $w\left(\sum_{i=1}^{n} x_i^2 + \lambda\right) = \sum_{i=1}^{n} x_i y_i$

bonus!

Why doesn't L2-Regularization set variables to 0?

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

Set equal to 0: $\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda w = 0$

re-arrange

$$w\Big(\underbrace{\sum_{i=1}^{n} x_i^2}_{\|x\|^2} + \lambda\Big) = \underbrace{\sum_{i=1}^{n} x_i y_i}_{y^T x}$$

Why doesn't L2-Regularization set variables to 0?

bonus!

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

Set equal to 0: $\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda w = 0$

re-arrange

$$w\left(\underbrace{\sum_{i=1}^{n} x_i^2}_{\|x\|^2} + \lambda\right) = \underbrace{\sum_{i=1}^{n} x_i y_i}_{y^Tx}$$

or $w = \frac{y^Tx}{\|x\|^2 + \lambda}$

- So as λ gets bigger, 'w' converges to 0.

Why doesn't L2-Regularization set variables to 0?

bonus!

- Consider an L2-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- Let's solve for the optimal 'w':

$$f'(w) = \sum_{i=1}^{n} x_i(wx_i - y_i) + \lambda w$$

Set equal to 0: $\sum_{i=1}^{n} x_i^2 w - \sum_{i=1}^{n} x_i y_i + \lambda w = 0$

re-arrange

$$w\Big(\underbrace{\sum_{i=1}^{n} x_i^2}_{\|x\|^2} + \lambda\Big) = \underbrace{\sum_{i=1}^{n} x_i y_i}_{y^Tx}$$

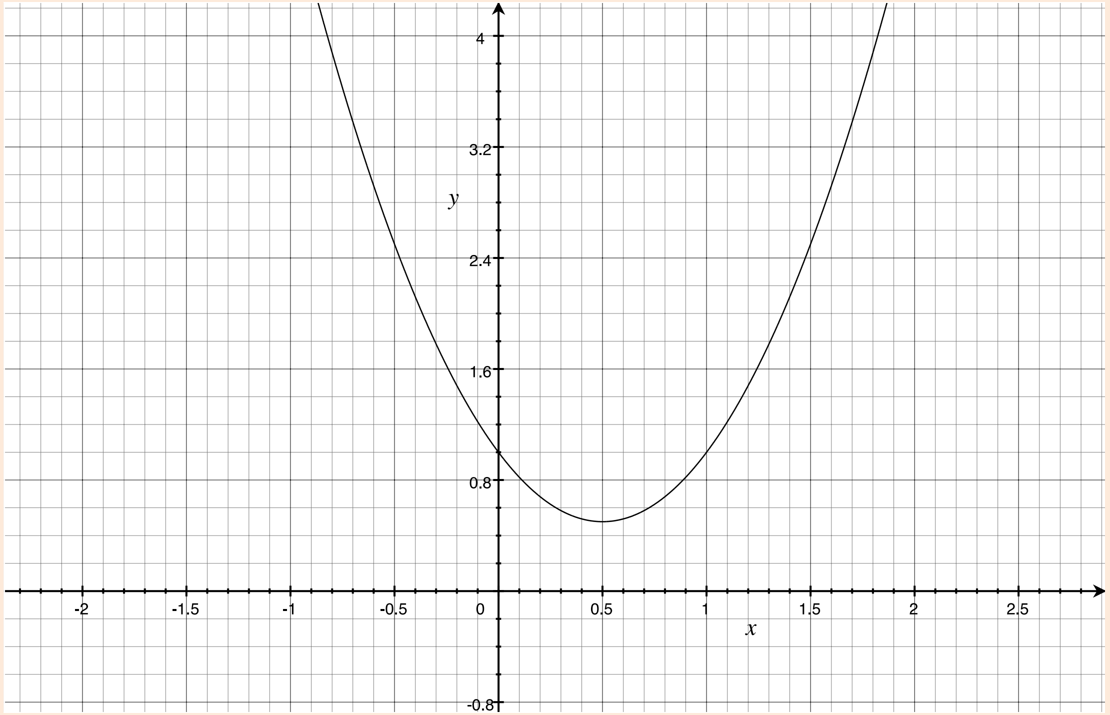
or $w = \dfrac{y^Tx}{\|x\|^2 + \lambda}$

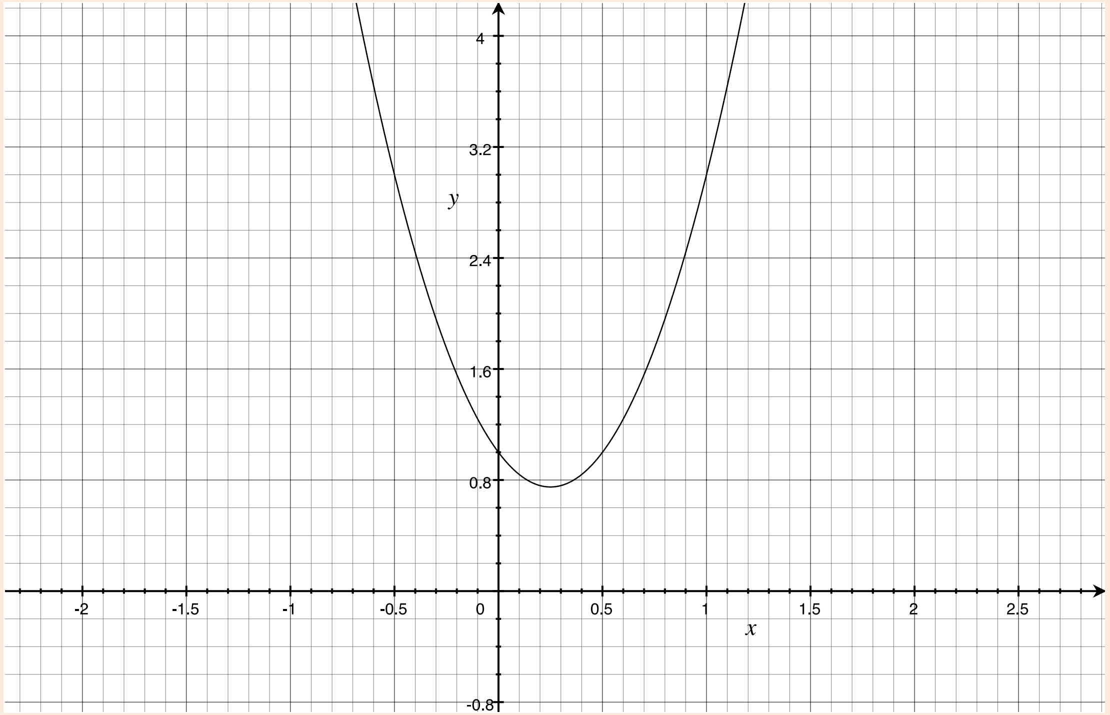
- So as λ gets bigger, 'w' converges to 0.
- However, for all finite λ 'w' will be non-zero unless $y^Tx = 0$ exactly.
  - But it's very unlikely that y^Tx will be exactly zero.

bonus!

Why doesn't L2-Regularization set variables to 0?

- Small λ — Big λ





- Solution further from zero — Solution closer to zero (but not exactly 0)

Why does L1-Regularization set things to 0?

bonus!

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

bonus!

Why does L1-Regularization set things to 0?

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then "left" limit and "right" limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda$$

$$f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

Why does L1-Regularization set things to 0?

bonus!

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then “left” limit and “right“ limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda$$

$$f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

- So which direction should “gradient descent” go in?

Why does L1-Regularization set things to 0?

bonus!

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then "left" limit and "right" limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda$$

$$f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

- So which direction should "gradient descent" go in?

$-f^-(0) = -y^Tx + \lambda$

$-f^+(0) = -y^Tx - \lambda$

Why does L1-Regularization set things to 0?

bonus!

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then "left" limit and "right" limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda$$

$$f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

- So which direction should "gradient descent" go in?

$$-f^-(0) = -y^Tx + \lambda$$
$$-f^+(0) = -y^Tx - \lambda$$

If these are positive $(-y^Tx > \lambda)$, we can improve by increasing 'w'.

bonus!

Why does L1-Regularization set things to 0?

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then "left" limit and "right" limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda$$

$$f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

- So which direction should "gradient descent" go in?

$$\left.\begin{aligned} -f^-(0) &= -y^Tx + \lambda \\ -f^+(0) &= -y^Tx - \lambda \end{aligned}\right\}$$ If these are positive $(-y^Tx > \lambda)$, we can improve by increasing 'w'.

$-y^Tx - \lambda$ in the second line has a brace under it: If these are negative $(y^Tx > \lambda)$, we can improve by decreasing 'w'.

Why does L1-Regularization set things to 0?

bonus!

- Consider an L1-regularized least squares problem with 1 feature:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \lambda|w|$$

- If (w = 0), then "left" limit and "right" limit are given by:

$$f^-(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) - \lambda = \sum_{i=1}^{n} x_i y_i - \lambda$$

$$f^+(0) = \sum_{i=1}^{n} x_i(0x_i - y_i) + \lambda = \sum_{i=1}^{n} x_i y_i + \lambda$$

- So which direction should "gradient descent" go in?

$-f^-(0) = -y^Tx + \lambda$ } If these are positive ($-y^Tx > \lambda$), we can improve by increasing 'w'.

$-f^+(0) = -y^Tx - \lambda$

If these are negative ($y^Tx > \lambda$), we can improve by decreasing 'w'.

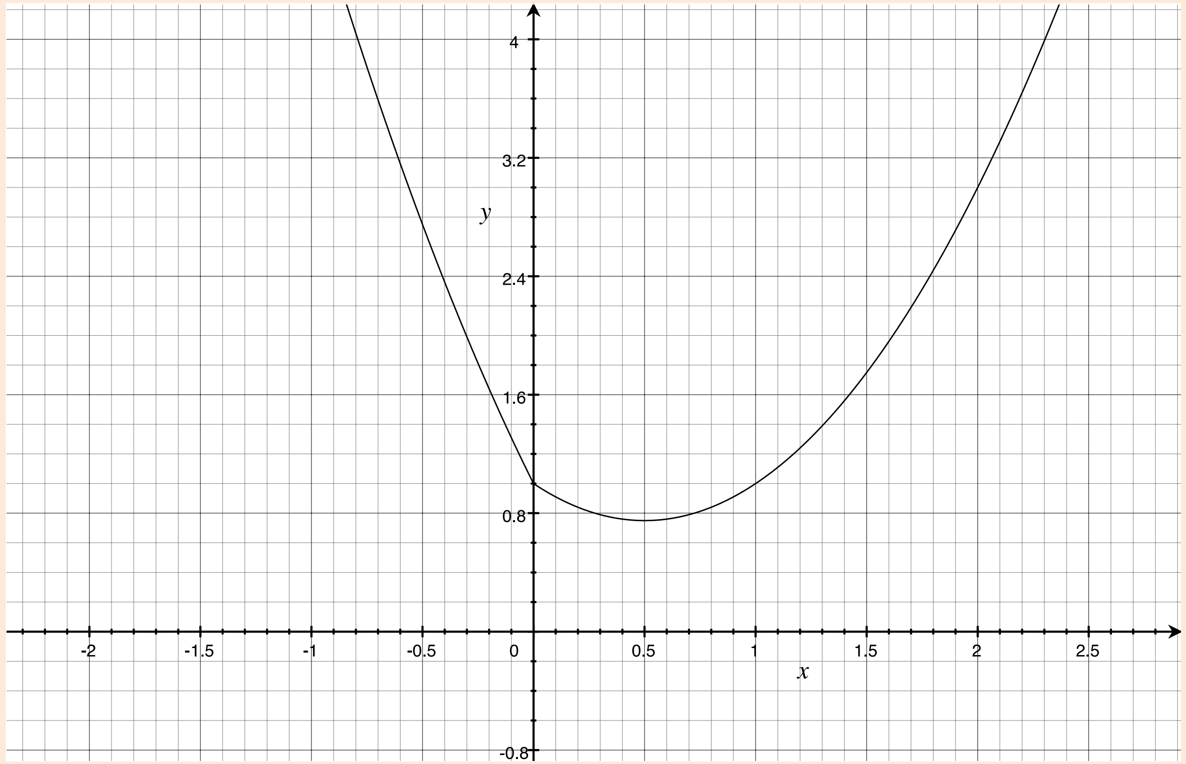
But if left and right "gradient descent" directions point in opposite directions ($|y^Tx| \leq \lambda$), minimum is 0.

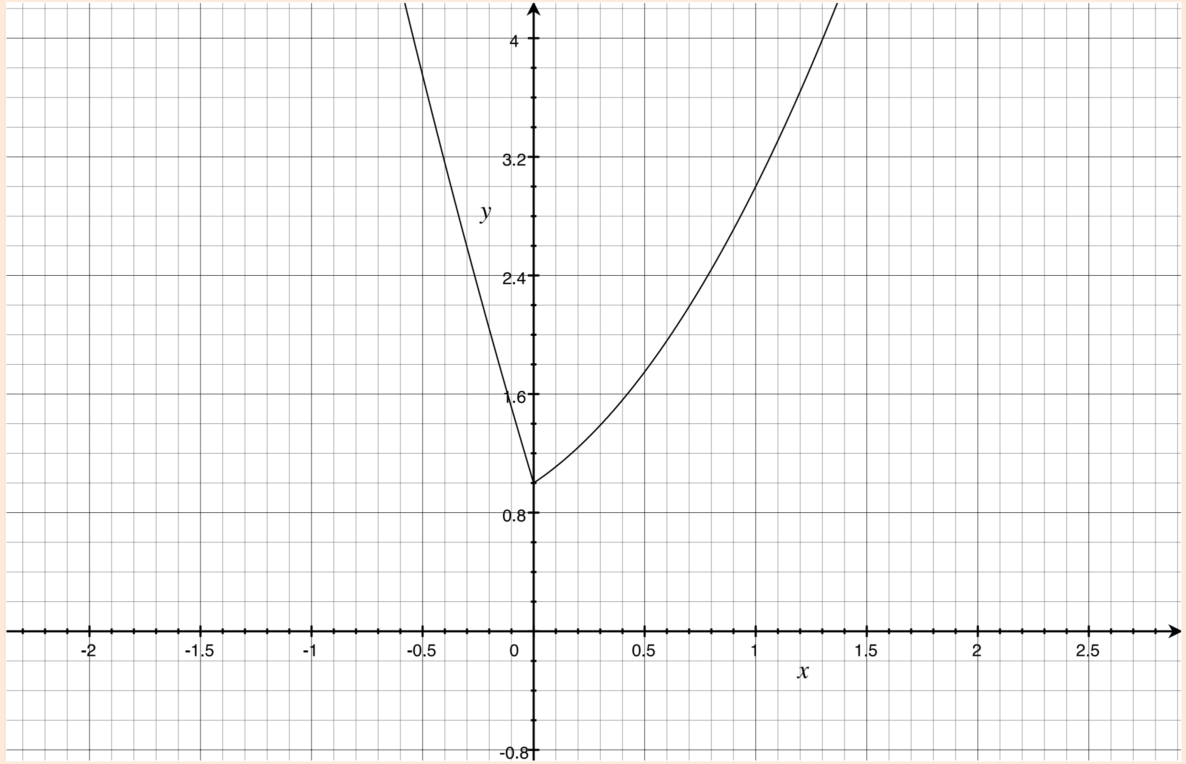
Why does L1-Regularization set things to 0?

bonus!

- Small λ

Big λ





- Solution nonzero

(minimum of left parabola is past origin, but right parabola is not)

Solution exactly zero

(minimum of both parabola are past the origin)

bonus!

L2-regularization vs. L1-regularization

- So with 1 feature:
  - L2-regularization only sets 'w' to 0 if $y^T x = 0$.
    - There is a only a single possible $y^T x$ value where the variable gets set to zero.
    - And λ has nothing to do with the sparsity.

bonus!

L2-regularization vs. L1-regularization

- So with 1 feature:
  - L2-regularization only sets ‘w’ to 0 if $y^Tx = 0$.
    - There is a only a single possible y^Tx value where the variable gets set to zero.
    - And λ has nothing to do with the sparsity.
  - L1-regularization sets ‘w’ to 0 if $|y^Tx| \leq \lambda$.
    - There is a range of possible y^Tx values where the variable gets set to zero.
    - And increasing λ increases the sparsity since the range of y^Tx grows.

bonus!

L2-regularization vs. L1-regularization

- So with 1 feature:
  - L2-regularization only sets 'w' to 0 if $y^Tx = 0$.
    - There is a only a single possible y^Tx value where the variable gets set to zero.
    - And λ has nothing to do with the sparsity.
  - L1-regularization sets 'w' to 0 if $|y^Tx| \leq \lambda$.
    - There is a range of possible y^Tx values where the variable gets set to zero.
    - And increasing λ increases the sparsity since the range of y^Tx grows.
- Note that it's important that the function is non-differentiable:
  - Differentiable regularizers penalizing size would need $y^Tx = 0$ for sparsity.

bonus!

L1-Loss vs. Huber Loss

- The same reasoning tells us the difference between the L1 *loss* and the Huber loss. They are very similar in that they both grow linearly far away from 0. So both are both robust but…
  - With the L1 loss the model often passes exactly through some points.
  - With Huber the model doesn't necessarily pass through any points.

- Why? With L1-regularization we were causing the elements of 'w' to be exactly 0. Analogously, with the L1-loss we cause the elements of 'r' (the residual) to be exactly zero. But zero residual for an example means you pass through that example exactly.

bonus!

Non-Uniqueness of L1-Regularized Solution

- How can L1-regularized least squares solution not be unique?
  - Isn't it convex?
- Convexity implies that minimum value of f(w) is unique (if exists), but there may be multiple 'w' values that achieve the minimum.

- Consider L1-regularized least squares with d=2, where feature 2 is a copy of a feature 1. For a solution (w_1,w_2) we have:

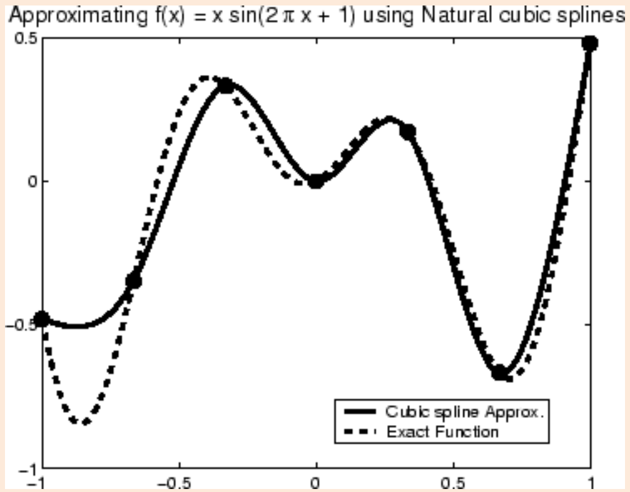
$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} = w_1 x_{i1} + w_2 x_{i1} = (w_1 + w_2) x_{i1}$$

- So we can get the same squared error with different w_1 and w_2 values that have the same sum. Further, if neither w_1 or w_2 changes sign, then $|w_1| + |w_2|$ will be the same so the new w_1 and w_2 will be a solution.

bonus!

Splines in 1D

- For 1D interpolation, alternative to polynomials/RBFs are splines:
  - Use a polynomial in the region between each data point.
  - Constrain some derivatives of the polynomials to yield a unique solution.
- Most common example is cubic spline:
  - Use a degree-3 polynomial between each pair of points.
  - Enforce that f'(x) and f''(x) of polynomials agree at all point.
  - "Natural" spline also enforces f''(x) = 0 for smallest and largest x.
- Non-trivial fact: natural cubic splines are sum of:
  - Y-intercept.
  - Linear basis.
  - RBFs with $g(\varepsilon) = \varepsilon^3$.
    - Different than Gaussian RBF because it *increases with distance*.

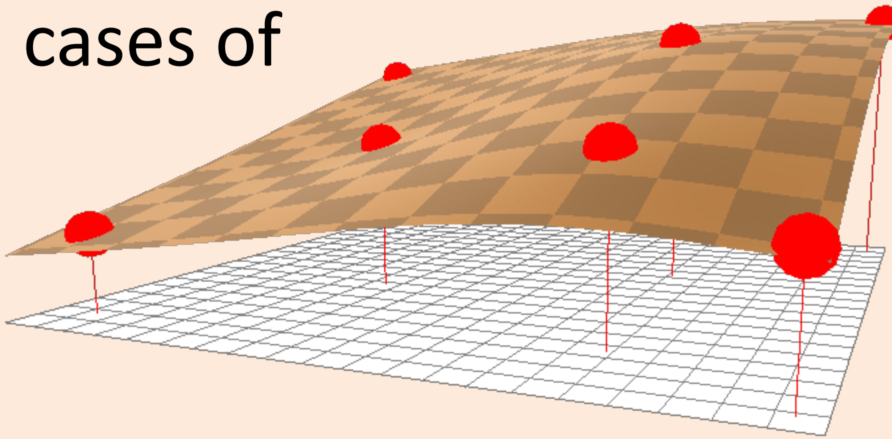


http://www.physics.arizona.edu/~restrepo/475A/Notes/sourcea-/node35.html

bonus!

Splines in Higher Dimensions

- Splines generalize to higher dimensions if data lies on a grid.
  - Many methods exist for grid-structured data (linear, cubic, splines, etc.).
  - For more general ("scattered") data, there isn't a natural generalization.
- Common 2D "scattered" data interpolation is thin-plate splines:
  - Based on curve made when bending sheets of metal.
  - Corresponds to RBFs with $g(\varepsilon) = \varepsilon^2 \log(\varepsilon)$.
- Natural splines and thin-plate splines: special cases of "polyharmonic" splines:
  - Less sensitive to parameters than Gaussian RBF.

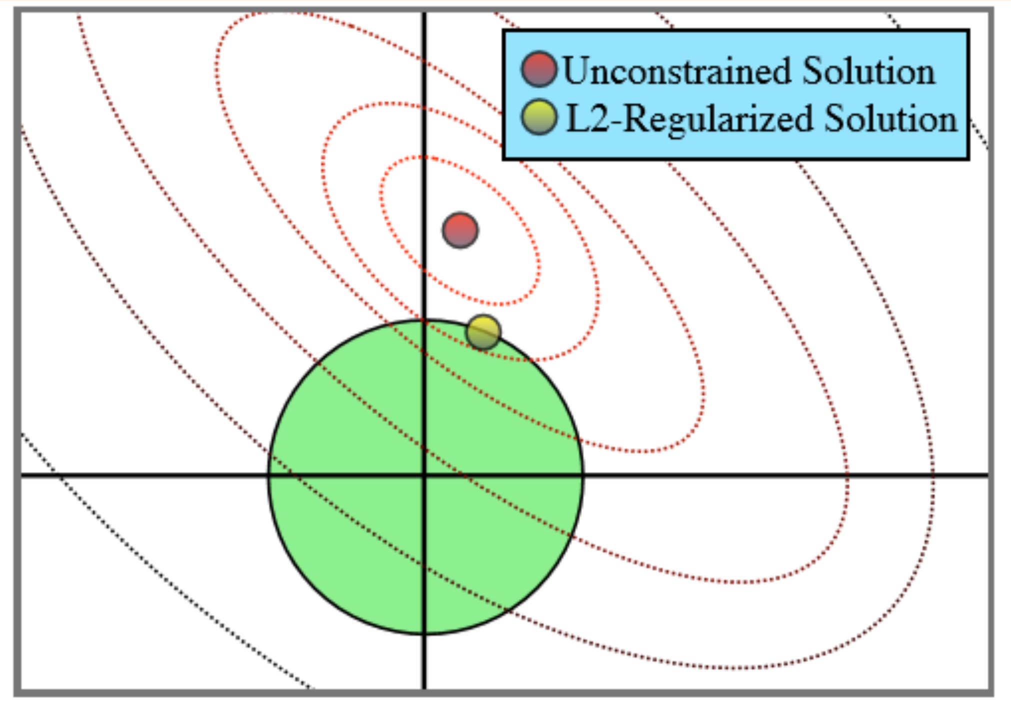


http://step.polymtl.ca/~rv101/thinplates/

bonus!

L2-Regularization vs. L1-Regularization

- L2-regularization conceptually restricts 'w' to a ball.



Minimizing $\frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$

is equivalent to minimizing

$\frac{1}{2}\|Xw - y\|^2$ subject to

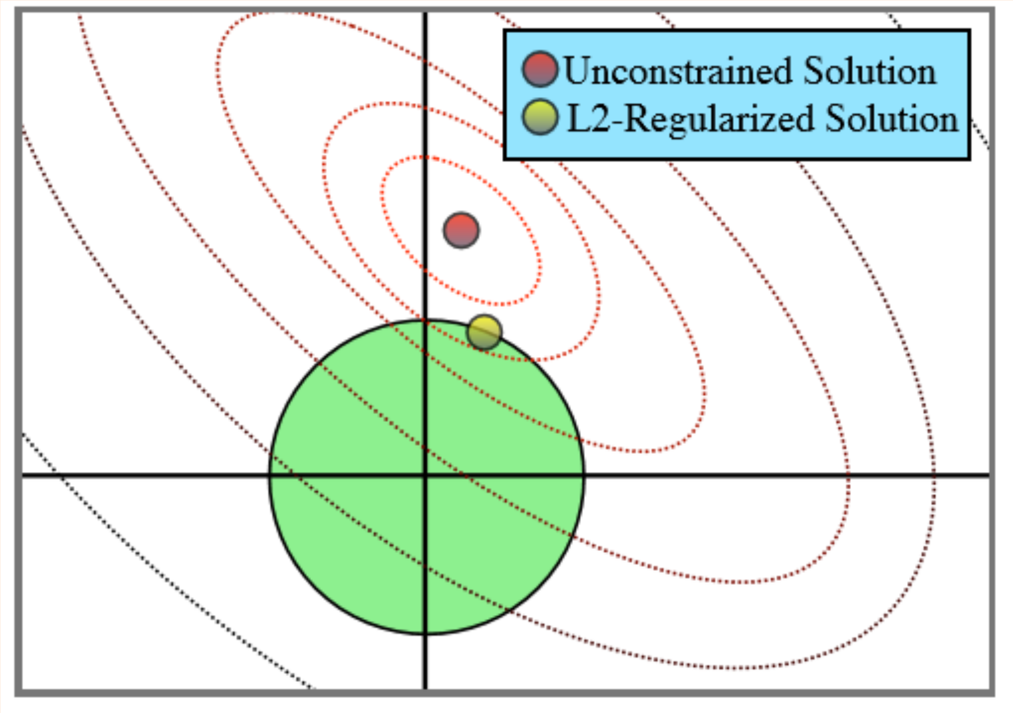
the constraint that $\|w\| \leq \tau$

for some value 'τ'

bonus!

L2-Regularization vs. L1-Regularization

- L2-regularization conceptually restricts 'w' to a ball.

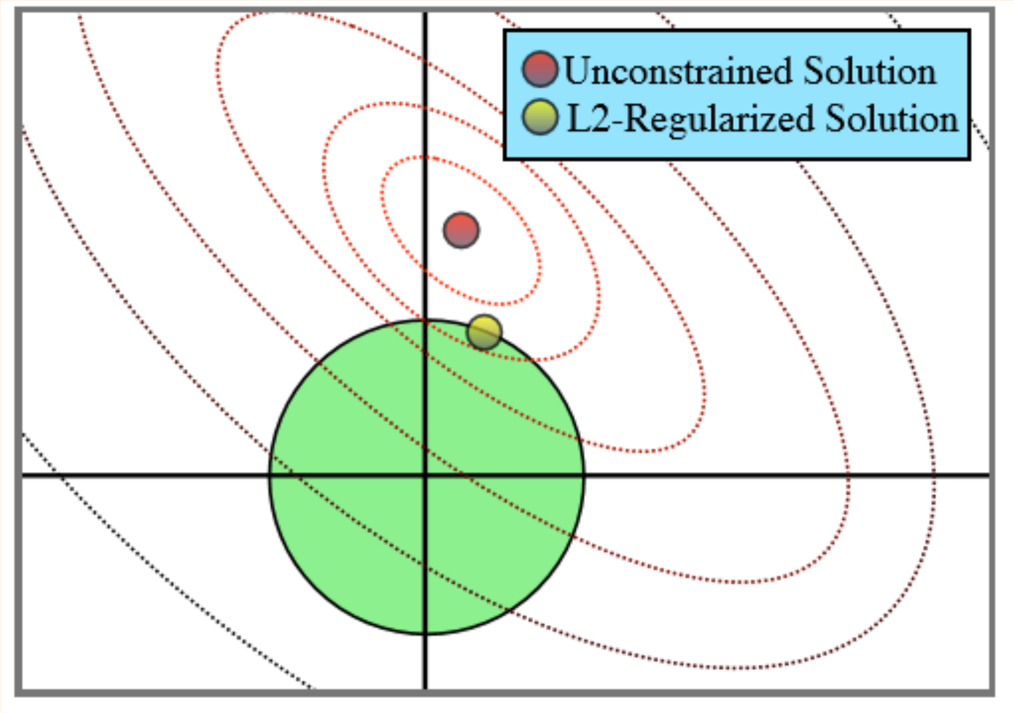


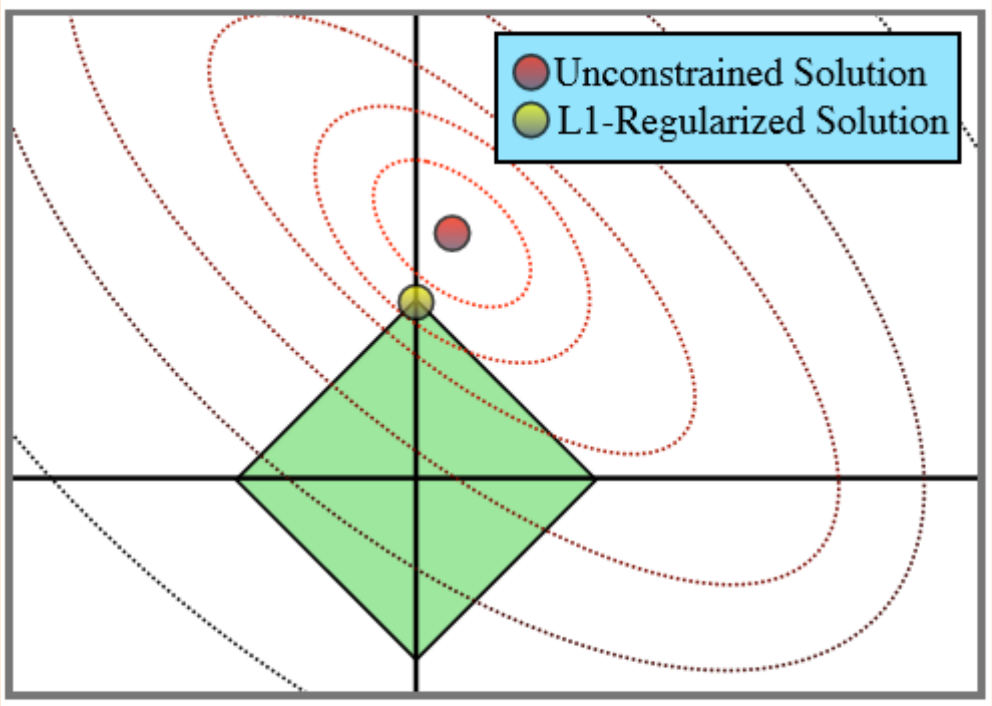
Related Infinite Series video

L2-Regularization vs. L1-Regularization

bonus!

- L2-regularization conceptually restricts ‘w’ to a ball.





- L1-regularization restricts to the L1 “ball”:
  - Solutions tend to be at corners where w_j are zero.

[Related Infinite Series video]