Trading off bias and variance, Cross-validation

Bias-variance tradeoff for least squares

If $Y_i = X_i^T w^* + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$

$$\widehat{w}_{\text{MLE}} = w^* + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon$$

$$\eta(x) = x^T w^*$$

$$\hat{f}_{\mathcal{D}}(x) = x^T w^* + x^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon$$

- Variance: $\mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]\right)^2\right] = \mathbb{E}_{\mathcal{D}}[x^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\epsilon^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}x]$
  $= \sigma^2\,\mathbb{E}_{\mathcal{D}}[x^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}x]$
  $= \sigma^2\,x^T\mathbb{E}_{\mathcal{D}}[(\mathbf{X}^T\mathbf{X})^{-1}]x$
- To analyze this, let's assume that $X_i \sim \mathcal{N}(0,\mathbf{I})$ and number of samples, n, is large enough such that $\mathbf{X}^T\mathbf{X} = n\mathbf{I}$ with high probability and $\mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}] \simeq \frac{1}{n}\mathbf{I}$, then
  - Variance is $\frac{\sigma^2 x^T x}{n}$, and decreases with increasing sample size n

Bias-Variance Properties of Ridge regression

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X, \;\; \mathbf{y} = \mathbf{X}w + \epsilon, \;\; \epsilon \sim \mathcal{N}(0,\sigma^2\mathbf{I})$
- The true error at a sample with feature x is

  $\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$

Bias-Variance Properties of Ridge regression

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X,\ \ \mathbf{y} = \mathbf{X}w + \epsilon,\ \ \epsilon \sim \mathcal{N}(0,\sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \underbrace{\mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x]}_{\text{Irreducible Error}} + \underbrace{\mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]}_{\text{Learning Error}}$$

Bias-Variance Properties of Ridge regression

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X,\ \mathbf{y} = \mathbf{X}w + \epsilon,\ \epsilon \sim \mathcal{N}(0,\sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\begin{aligned}
&\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x] \\
&\quad = \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x] \\
&\quad = \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]
\end{aligned}$$

Bias-Variance Properties of Ridge regression

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X, \;\; \mathbf{y} = \mathbf{X}w + \epsilon, \;\; \epsilon \sim \mathcal{N}(0,\sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \sigma^2 + (x^Tw - \mathbb{E}_{\mathcal{D}_{\text{train}}}[x^T\hat{w}_{\text{ridge}}\,|\,x])^2 + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}_{\tilde{\mathcal{D}}_{\text{train}}}[x^T\hat{w}_{\text{ridge}}\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

Irreduc. Error — Bias-squared — Variance

Bias-Variance Properties of Ridge regression

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X,\ \ \mathbf{y} = \mathbf{X}w + \epsilon,\ \ \epsilon \sim \mathcal{N}(0,\sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \mathbb{E}_{y|x}[(y - x^Tw)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^Tw - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

$$= \sigma^2 + (x^Tw - \mathbb{E}_{\mathcal{D}_{\text{train}}}[x^T\hat{w}_{\text{ridge}}\,|\,x])^2 + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}_{\tilde{\mathcal{D}}_{\text{train}}}[x^T\hat{w}_{\text{ridge}}\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]$$

Irreduc. Error — Bias-squared — Variance

Suppose $\mathbf{X}^T\mathbf{X} = n\mathbf{I}$, then $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T(\mathbf{X}w + \epsilon)$

$$= \frac{n}{n+\lambda}w + \frac{1}{n+\lambda}\mathbf{X}^T\epsilon$$

Bias-Variance Properties

Suppose $\mathbf{X}^T\mathbf{X} = n\mathbf{I}$, then

$$\hat{w}_{\text{ridge}} = \frac{n}{n+\lambda} w + \frac{1}{n+\lambda} \mathbf{X}^T \epsilon$$

- Recall: $\hat{w}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
- To analyze bias-variance tradeoff, we need to assume probabilistic generative model: $x_i \sim P_X, \;\; \mathbf{y} = \mathbf{X}w + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
- The true error at a sample with feature x is

$$\begin{aligned}
&\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T\hat{w}_{\text{ridge}})^2 \,|\, x] \\
&\quad = \mathbb{E}_{y|x}[(y - \mathbb{E}[y\,|\,x])^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}[y\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x] \\
&\quad = \mathbb{E}_{y|x}[(y - x^T w)^2 \,|\, x] + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(x^T w - x^T\hat{w}_{\text{ridge}})^2 \,|\, x] \\
&= \sigma^2 + (x^T w - \mathbb{E}_{\mathcal{D}_{\text{train}}}[x^T\hat{w}_{\text{ridge}}\,|\,x])^2 + \mathbb{E}_{\mathcal{D}_{\text{train}}}[(\mathbb{E}_{\tilde{\mathcal{D}}_{\text{train}}}[x^T\hat{w}_{\text{ridge}}\,|\,x] - x^T\hat{w}_{\text{ridge}})^2 \,|\, x]
\end{aligned}$$

(verify at home)

$$= \sigma^2 + \frac{\lambda^2}{(n+\lambda)^2}(w^T x)^2 + \frac{\sigma^2 n}{(n+\lambda)^2}\|x\|_2^2$$

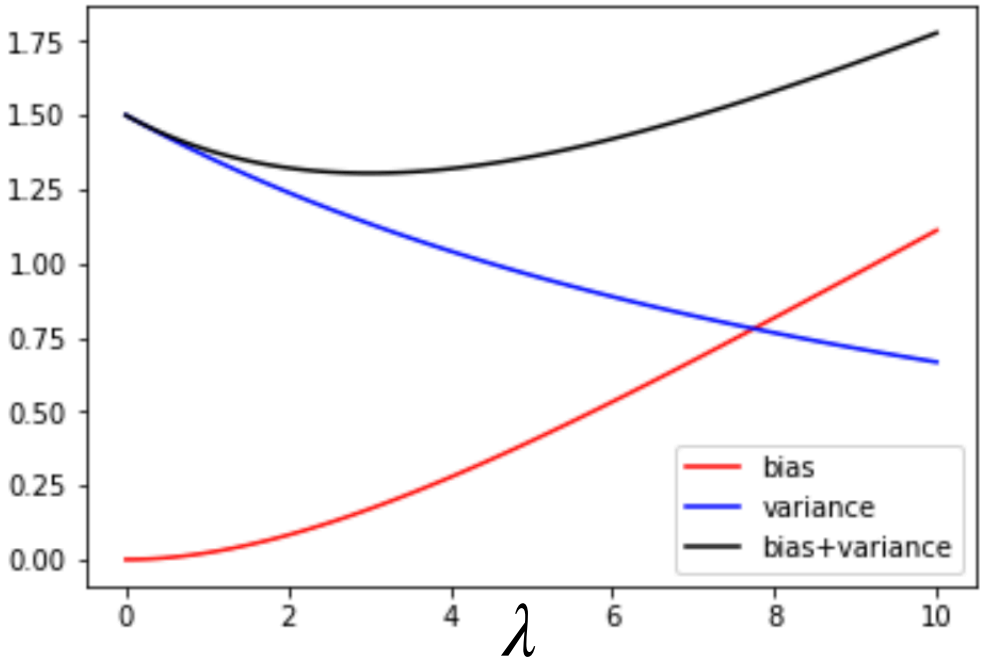
Irreduc. Error | Bias-squared | Variance

Bias-Variance Properties of Ridge regression

- Ridge regressor: $\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$
- True error

$$\mathbb{E}_{y,\mathcal{D}_{\text{train}}|x}[(y - x^T \hat{w}_{\text{ridge}})^2 \,|\, x] = \sigma^2 + \underbrace{\frac{\lambda^2}{(n+\lambda)^2}(w^T x)^2}_{\text{Bias-squared}} + \underbrace{\frac{\sigma^2 n}{(n+\lambda)^2}\|x\|_2^2}_{\text{Variance}}$$

d=10, n=20, $\sigma^2 = 3.0, \|w\|_2^2 = 10$



as $\lambda \to 0$, $\hat{w}_{\text{ridge}} \to \hat{w}_{\text{LS}}$

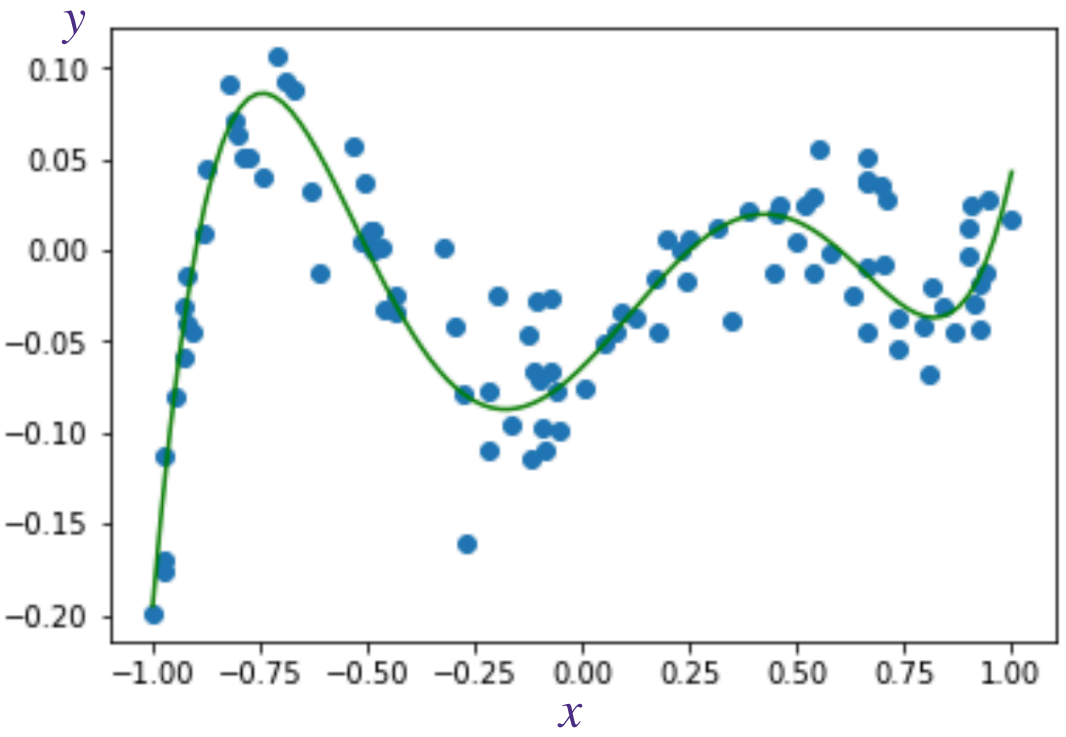
as $\lambda \to \infty$ $\hat{w}_{\text{ridge}} \to 0$

What you need to know…

- Regularization
  - Penalizes complex models towards preferred, simpler models
- Ridge regression
  - L_2 penalized least-squares regression
  - Regularization parameter trades off model complexity with training error
  - Never regularize the offset!

Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

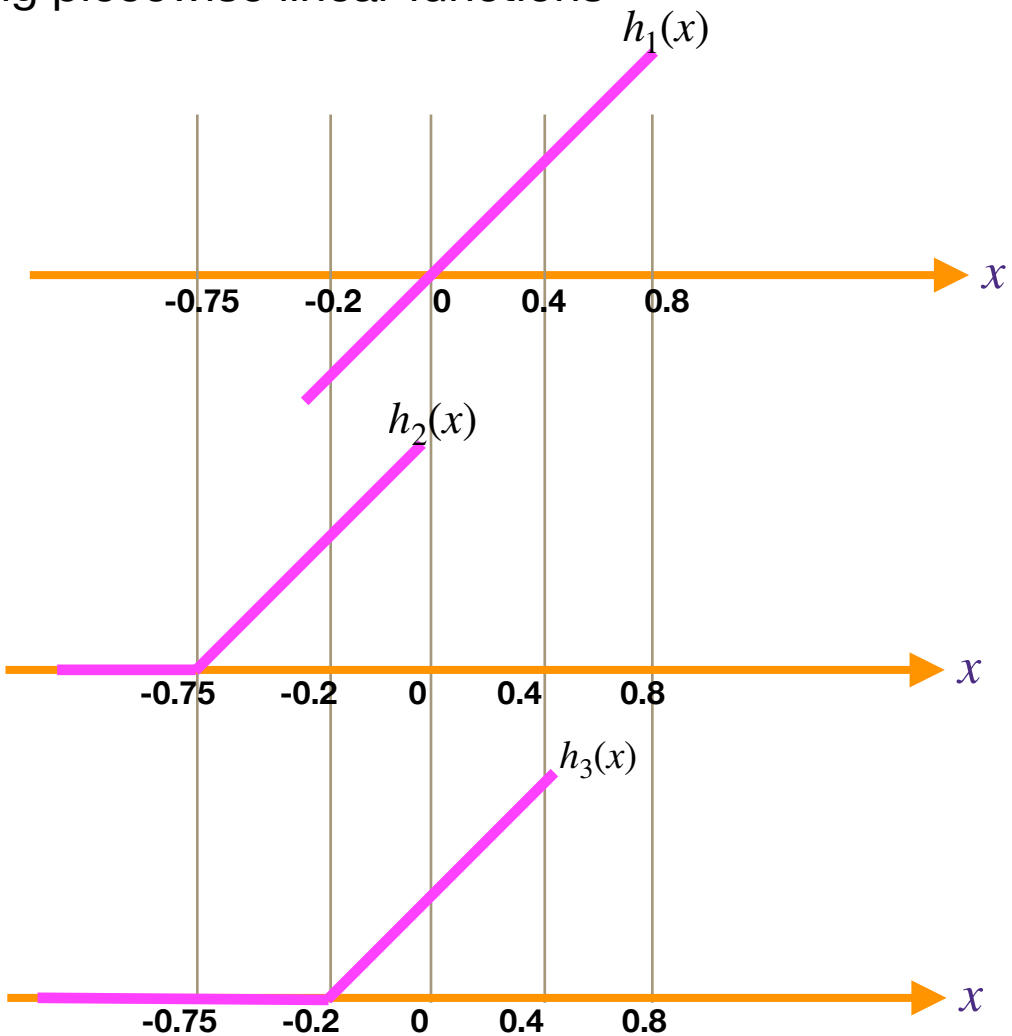


Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x + 0.75]^+ \\ [x + 0.2]^+ \\ [x - 0.4]^+ \\ [x - 0.8]^+ \end{bmatrix}$$

$$[a]^+ \triangleq \max\{a, 0\}$$

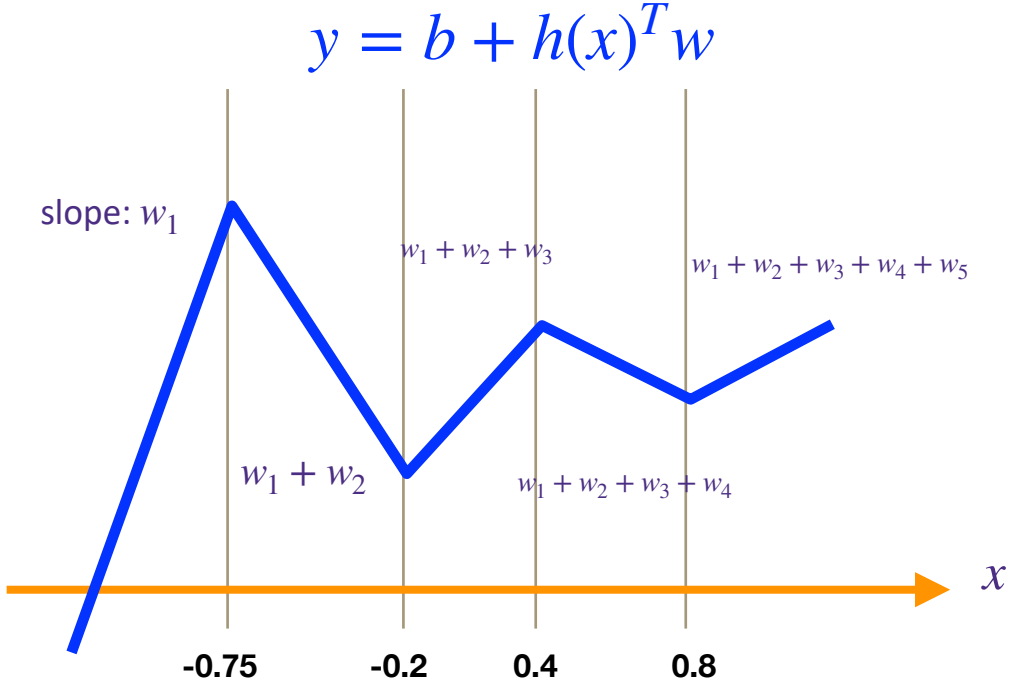


Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x + 0.75]^+ \\ [x + 0.2]^+ \\ [x - 0.4]^+ \\ [x - 0.8]^+ \end{bmatrix}$$

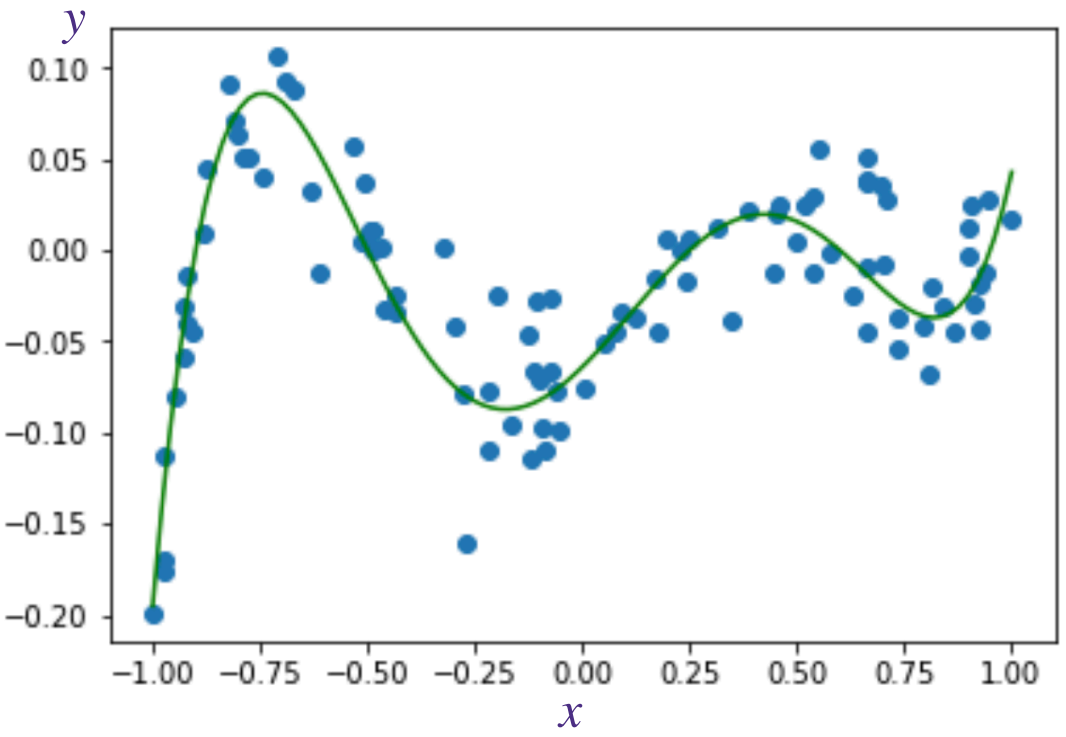
$$[a]^+ \triangleq \max\{a, 0\}$$



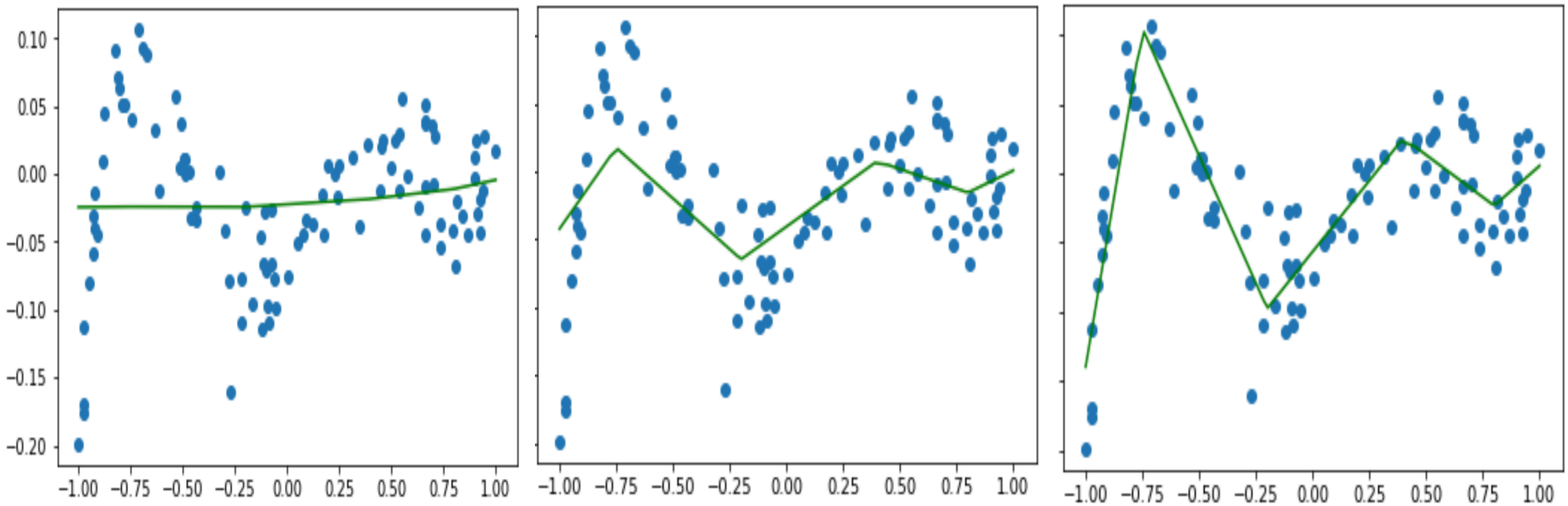
the weights capture the change in the slopes

Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions



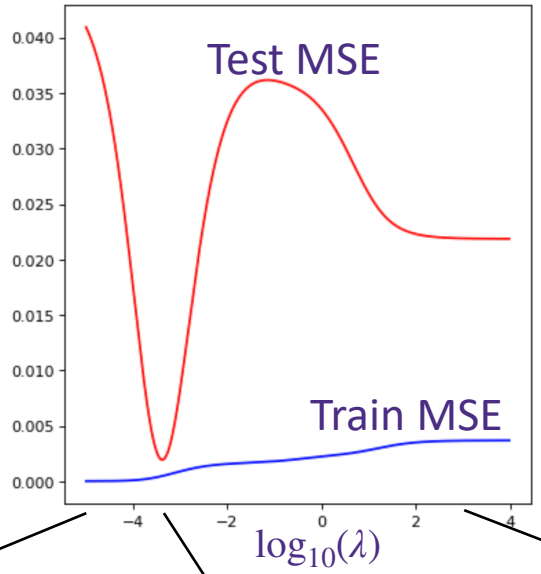
Example: piecewise linear fit (ridge regression)

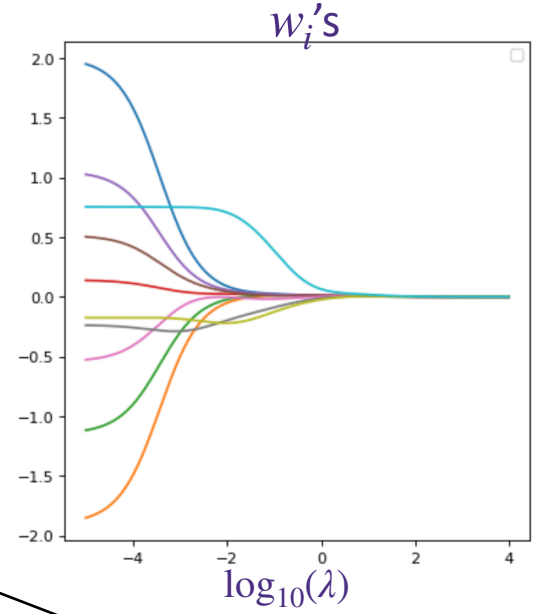


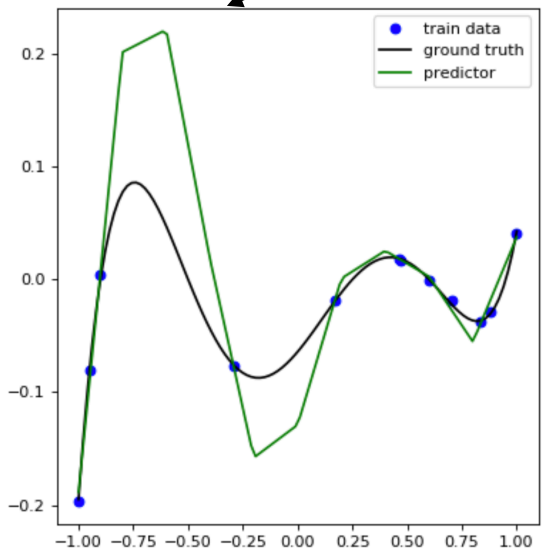
$\lambda = 1$ $\lambda = 0.005$ $\lambda = 0.000001$

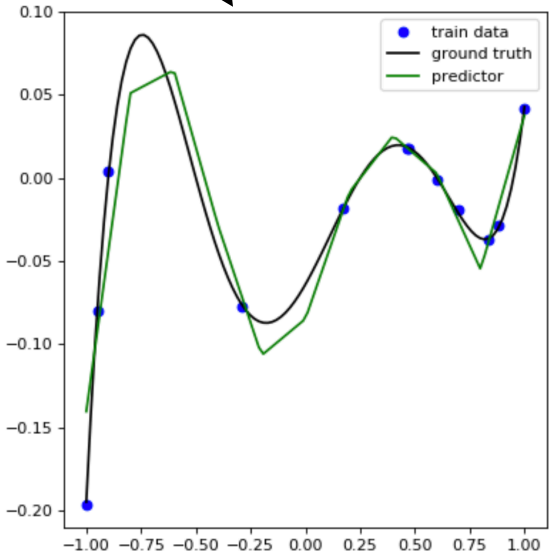
We do not observe overfitting, as d=5 and n=100

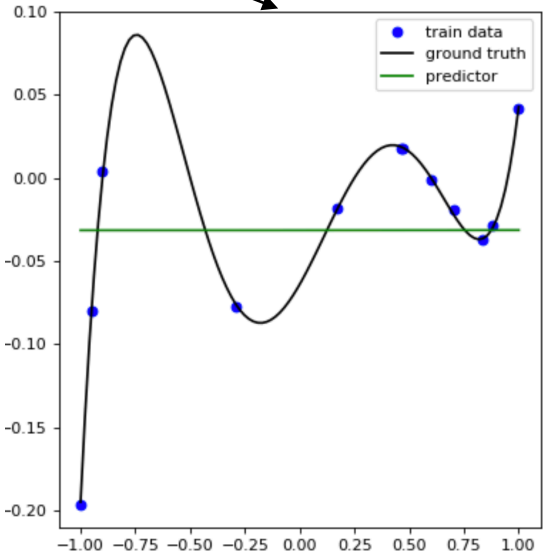
Piecewise linear with $w \in \mathbb{R}^{10}$ and n=11 samples









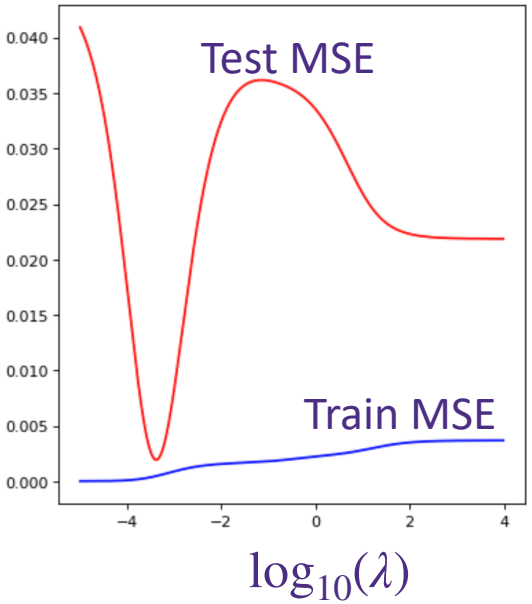


Model selection using Cross-validation

How… How… How???????

- Ridge regression:
  How do we pick the regularization constant λ…
- Polynomial features:
  How do we pick the number of basis functions…

- We could use the test data, but…



How… How… How????????

- Ridge regression:
  How do we pick the regularization constant λ…
- Polynomial features:
  How do we pick the number of basis functions…

- We could use the test data, but…
  - Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever **train on the test data**
  - Use test data only for reporting the test error
    (once in the end)

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - $\mathscr{D}$: training data
  - $\mathscr{D} \setminus j$: training data with j-th data point (x_j, y_j) moved to validation set
- Learn model $f_{\mathscr{D} \setminus j}$ with $\mathscr{D} \setminus j$ dataset
- The squared error on predicting y_j: $(y_j - f_{\mathscr{D} \setminus j}(x_j))^2$

  is an unbiased estimate of the **true error**

$$\text{error}_{\text{true}}(f_{\mathscr{D} \setminus j}) = \mathbb{E}_{(x,y) \sim P_{x,y}}[(y - f_{\mathscr{D} \setminus j}(x))^2]$$

  but, variance of $(y_j - f_{\mathscr{D} \setminus j}(x_j))^2$ is too large

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - $\mathscr{D}$: training data
  - $\mathscr{D} \setminus j$: training data with j-th data point (x_j, y_j) moved to validation set
- Learn model $f_{\mathscr{D}\setminus j}$ with $\mathscr{D} \setminus j$ dataset
- The squared error on predicting y_j: $(y_j - f_{\mathscr{D}\setminus j}(x_j))^2$
  is an unbiased estimate of the **true error**

$$\text{error}_{\text{true}}(f_{\mathscr{D}\setminus j}) = \mathbb{E}_{(x,y)\sim P_{x,y}}[(y - f_{\mathscr{D}\setminus j}(x))^2]$$

  but variance of $(y_j - f_{\mathscr{D}\setminus j}(x_j))^2$ is too large, so instead

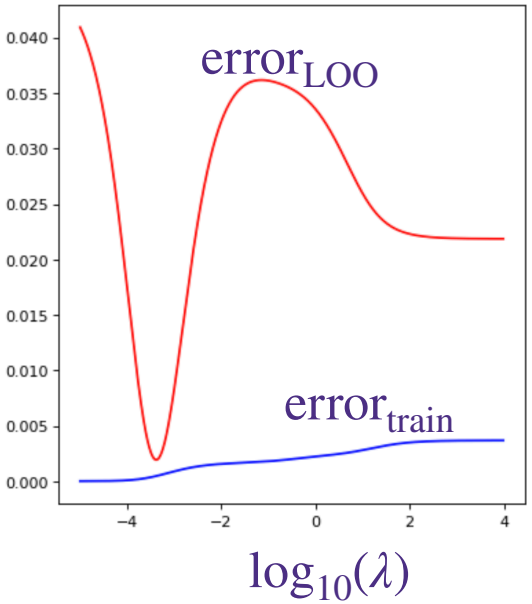
- **LOO cross validation**: Average over all data points *j*:
  - Train n times:
    for each data point you leave out, learn a new classifier $f_{\mathscr{D}\setminus j}$
  - **Estimate the true error** as:

$$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{\mathcal{D}\setminus j}(x_j))^2$$

LOO cross validation is (almost) unbiased estimate!

- When computing LOOCV error, we only use $n-1$ data points to train
  - So it's not estimate of true error of learning with n data points
  - Usually pessimistic – learning with less data typically gives worse answer. (Leads to an over estimation of the error)

- LOO is almost unbiased! Use LOO error for model selection!!!
  - **E.g., picking λ**



Computational cost of LOO

- Suppose you have 100,000 data points
- say, you implemented a fast version of your learning algorithm
  - Learns in only 1 second
- Computing LOO will take about 1 day!!

Use *k*-fold cross validation

- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{\mathcal{D}\setminus\mathcal{D}_i}$ using data point not in $\mathcal{D}_i$
  - Estimate error of $f_{\mathcal{D}\setminus\mathcal{D}_i}$ on validation set $\mathcal{D}_i$:

$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\setminus\mathcal{D}_i}(x_j))^2$$

$\mathcal{D} =$

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$
Train	Train	Validation	Train	Train

$f_{\mathcal{D}\setminus\mathcal{D}_3}$

Use *k*-fold cross validation

- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{\mathcal{D}\setminus\mathcal{D}_i}$ using data point not in $\mathcal{D}_i$
  - Estimate error of $f_{\mathcal{D}\setminus\mathcal{D}_i}$ on validation set $\mathcal{D}_i$:

$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\setminus\mathcal{D}_i}(x_j))^2$$

- k-fold cross validation error is average over data splits:

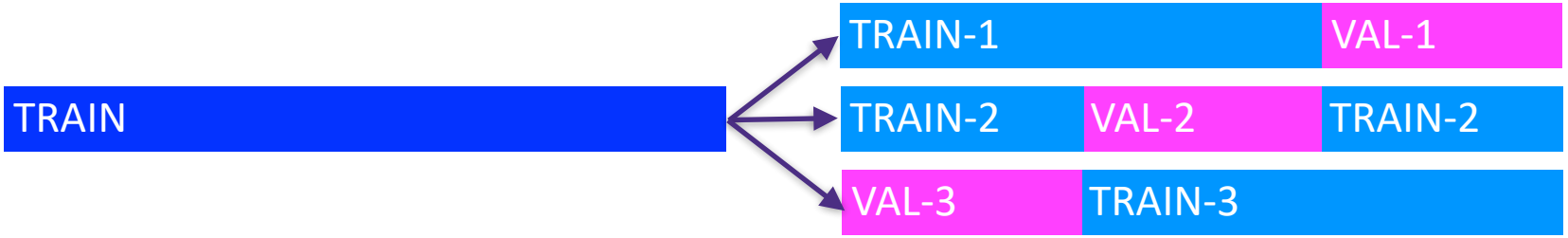
$$\text{error}_{k-\text{fold}} = \frac{1}{k}\sum_{i=1}^{k} \text{error}_{\mathcal{D}_i}$$

- k-fold cross validation properties:
  - Much faster to compute than LOO as $k \ll n$
  - More (pessimistically) biased – using much less data, only $n - \frac{n}{k}$
  - Usually, k = 10

Recap

- Given a dataset, begin by splitting into

- Model selection: Use k-fold cross-validation on TRAIN to train predictor and choose hyper-parameters such as λ



- Model assessment: Use TEST to assess the accuracy of the model you output
  - Never ever ever ever ever train or choose parameters based on the test data

Model selection using cross validation

- **For** $\lambda \in \{0.001, 0.01, 0.1, 1, 10\}$
  - **For** $j \in \{1, \ldots, k\}$
    - $$\hat{w}_{\lambda,\text{Train}-j} \leftarrow \arg\min_{w} \sum_{i \in \text{Train}-j} (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$$
- $$\hat{\lambda} \leftarrow \arg\min_{\lambda} \frac{1}{k} \sum_{j=1}^{k} \sum_{i \in \text{Val}-j} (y_i - \hat{w}_{\lambda,\text{Train}-j}^T x_i)^2$$

Example 1

- You wish to predict the stock price of zoom.us given historical stock price data y_i's (for each i-th day) and the historical news articles x_i's
- You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST
- What's wrong with this procedure?

Example 2

- Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the training set:

50 indices j that have largest $\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$

- After picking our 50 features, we then use CV with the training set to train ridge regression with regularization λ
- What's wrong with this procedure?

Recap

- Learning is…
  - Collect some data
    - E.g., housing info and sale price
  - Randomly split dataset into TRAIN, VAL, and TEST
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear transformations
  - Choose a loss function
    - E.g., least squares with ridge regression penalty on TRAIN
  - **Choose an optimization procedure**
    - **E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features and amount of regularization**
  - Justifying the accuracy of the estimate
    - E.g., report TEST error

Simple variable selection: LASSO for sparse regression

Sparsity

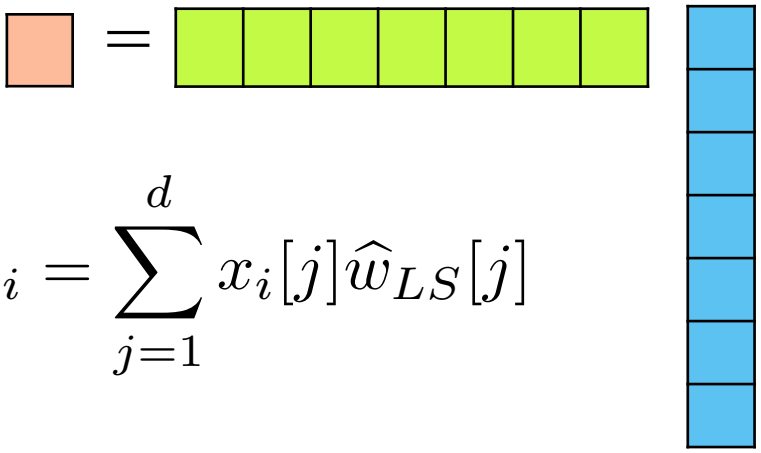
$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector w is **sparse**, if many entries are zero

Sparsity

$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector w is **sparse**, if many entries are zero
  - **Efficiency**: If size(w) = 100 Billion, each prediction $w^T x$ is expensive:
    - If w is sparse, prediction computation only depends on number of non-zeros in w



$$\widehat{y}_i = \widehat{w}_{LS}^{\top} x_i = \sum_{j=1}^{d} x_i[j]\widehat{w}_{LS}[j]$$

Sparsity

$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector w is **sparse**, if many entries are zero
  - **Interpretability**: What are the relevant features to make a prediction?

Lot size
Single Family
Year built
Last sold price
Last sale price/sqft
Finished sqft
Unfinished sqft
Finished basement sqft
floors
Flooring types
Parking type
Parking amount
Cooling
Heating
Exterior materials
Roof type
Structure style

Dishwasher
Garbage disposal
Microwave
Range / Oven
Refrigerator
Washer
Dryer
Laundry location
Heating type
Jetted Tub
Deck
Fenced Yard
Lawn
Garden
Sprinkler System

- How do we find "best" subset of features useful in predicting the price among all possible combinations?

Finding best subset: Exhaustive

- Try all subsets of size 1, 2, 3, … and one that minimizes validation error
- Problem?

Finding best subset: Greedy

Forward stepwise:
Starting from simple model and iteratively add features most useful to fit

Backward stepwise:
Start with full model and iteratively remove features least useful to fit

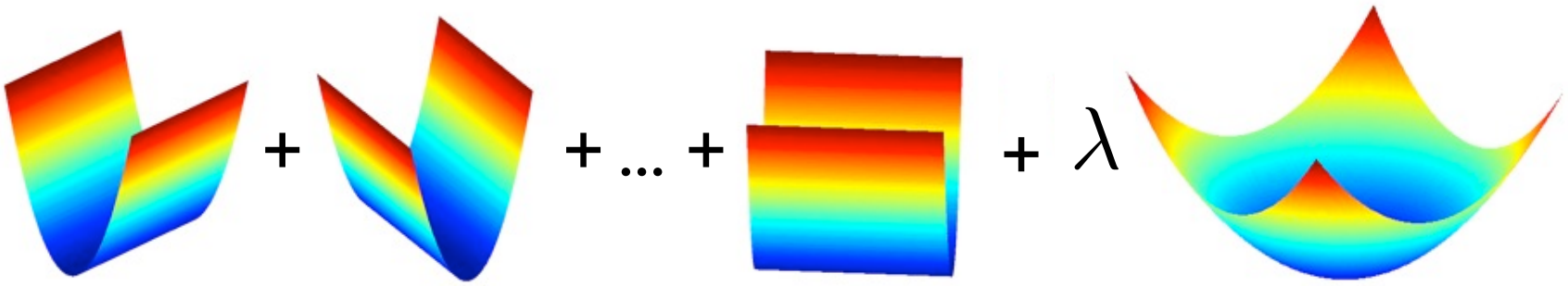
Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Finding best subset: Regularize

Ridge regression makes coefficients small

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

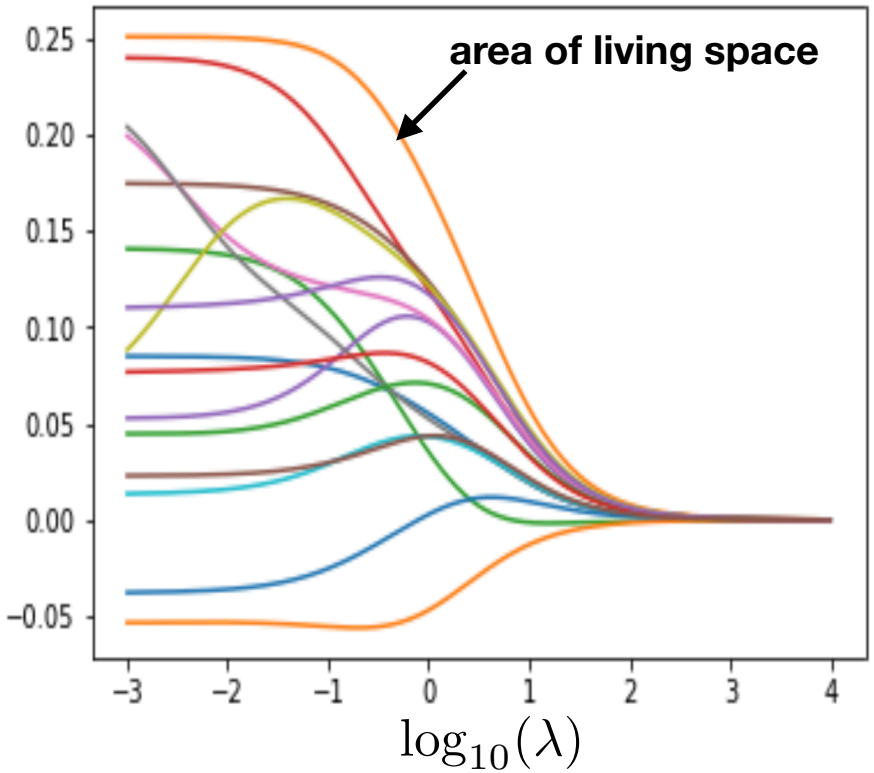


Finding best subset: Regularize

Ridge regression makes coefficients small

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

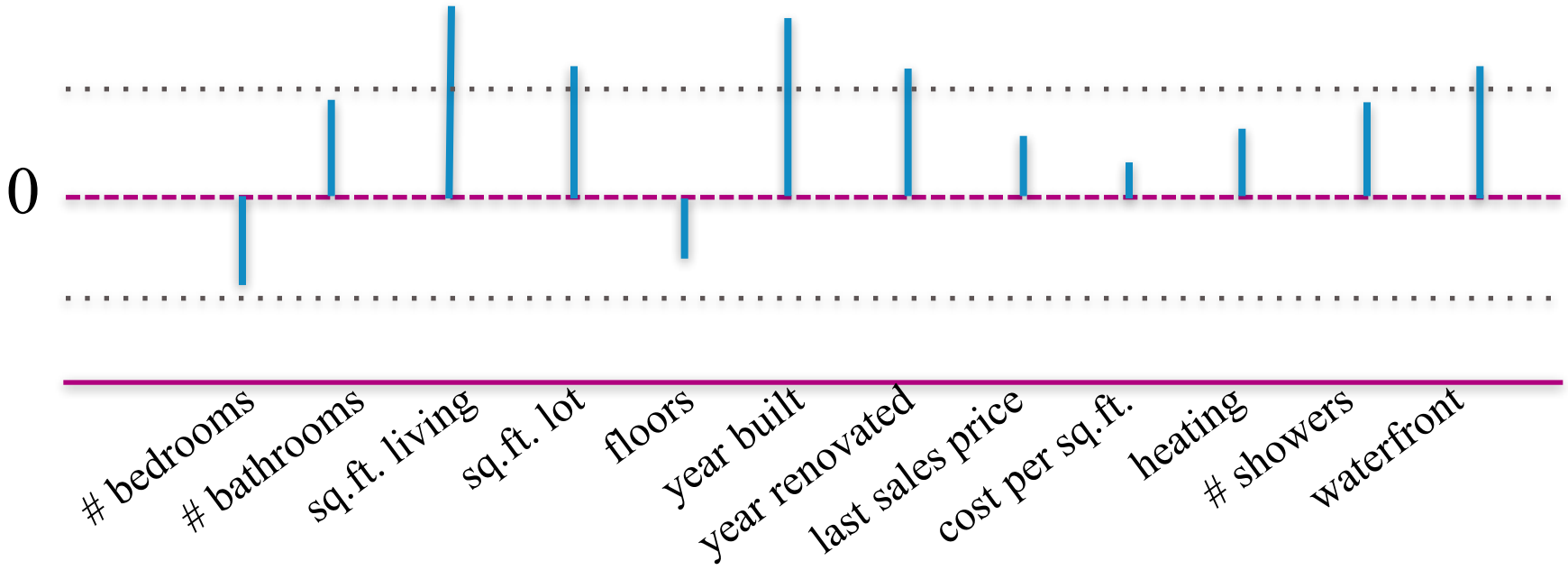
w_i's



Thresholded Ridge Regression

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

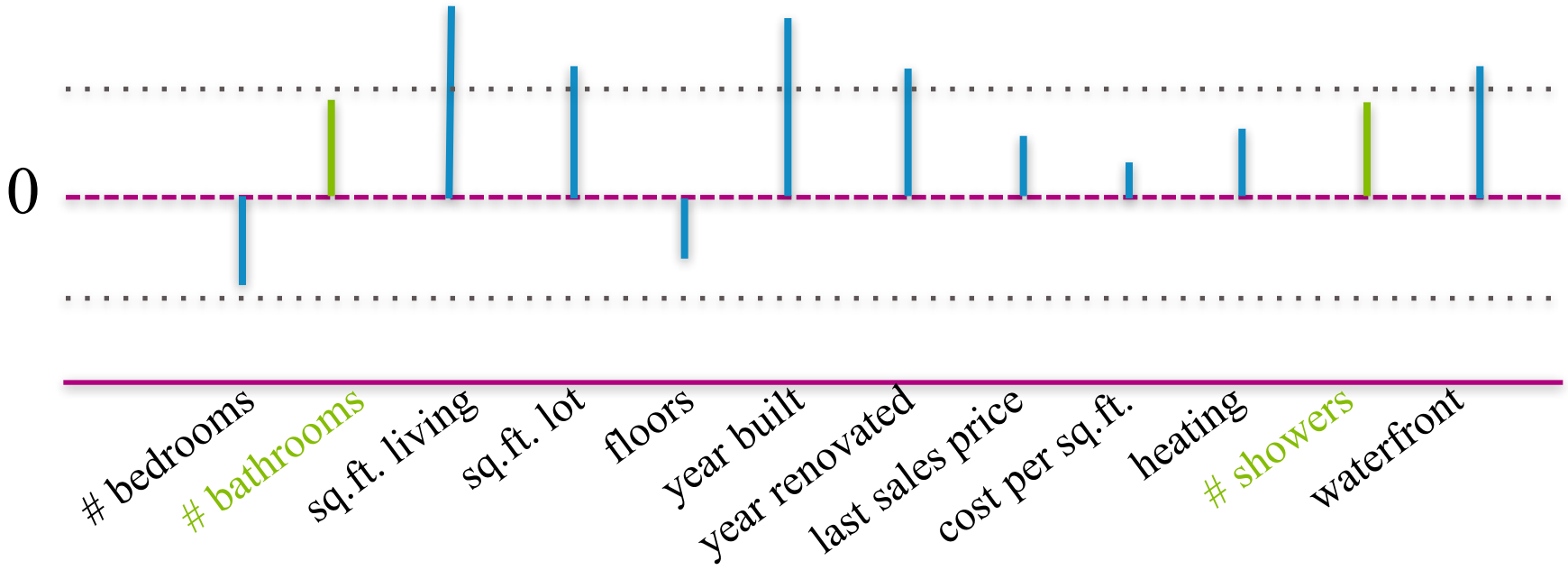
Why don't we just set **small** ridge coefficients to 0?



Thresholded Ridge Regression

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

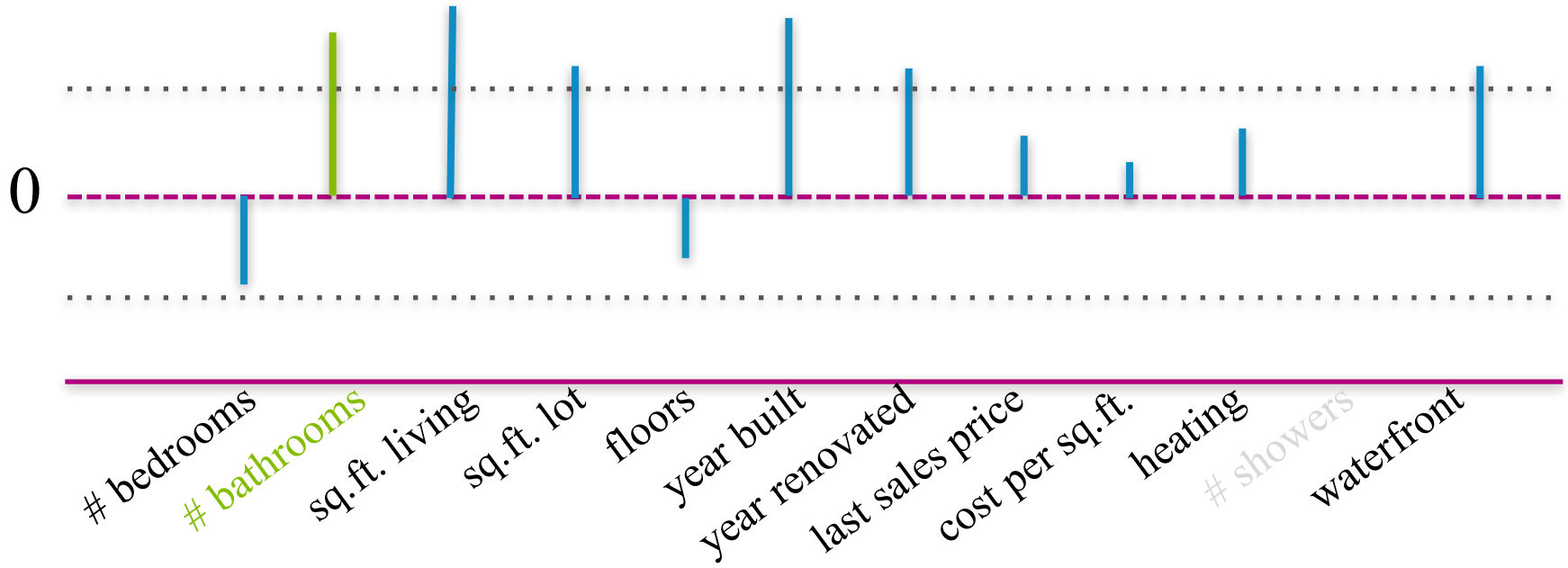
Consider two related features (bathrooms, showers)



Thresholded Ridge Regression

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

What if we **didn't** include showers? Weight on bathrooms increases!

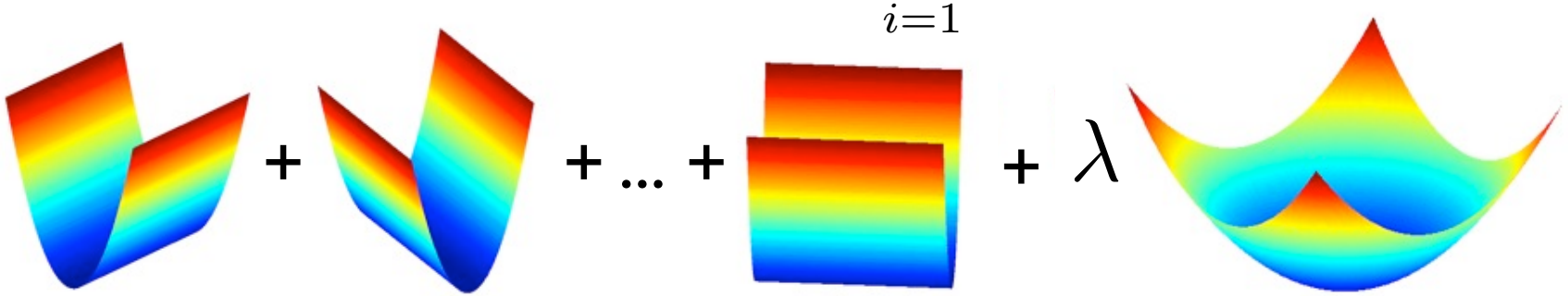


Can another regularizer perform selection automatically?

Recall Ridge Regression

- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

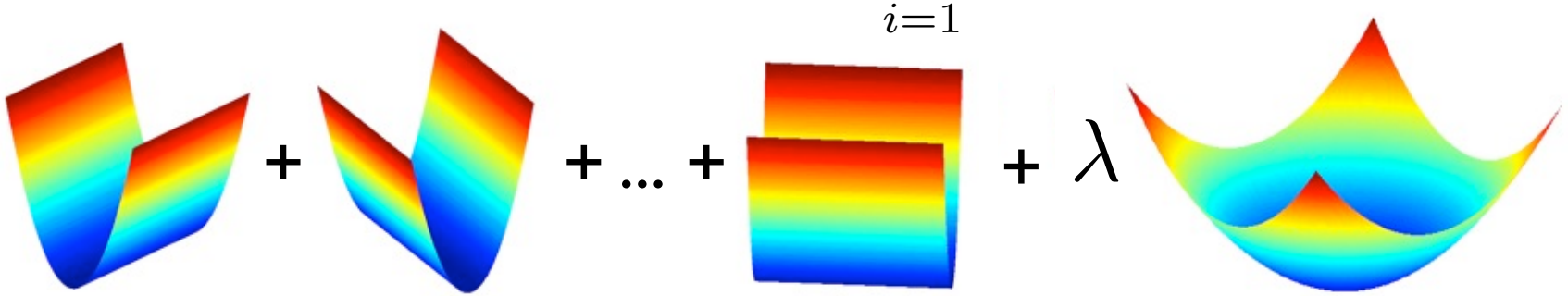


$$\|w\|_p = \left(\sum_{i=1}^{d} |w|^p\right)^{1/p}$$

Ridge vs. Lasso Regression

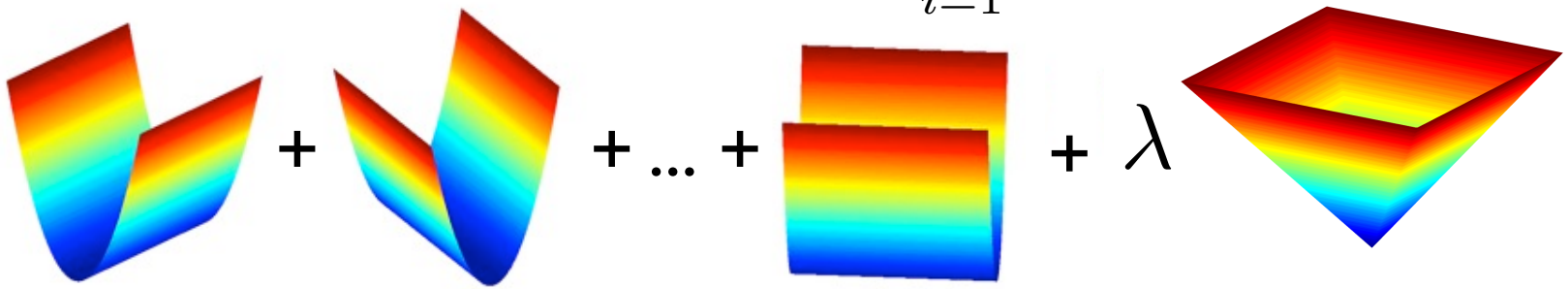
- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



- Lasso objective:

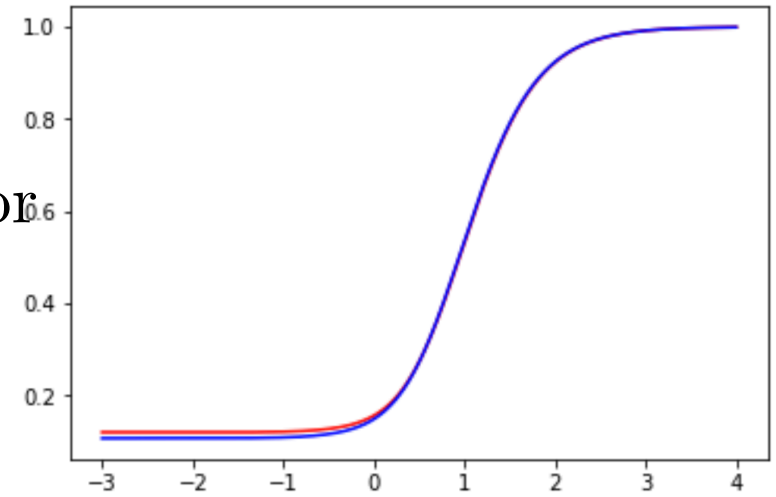
$$\widehat{w}_{lasso} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$

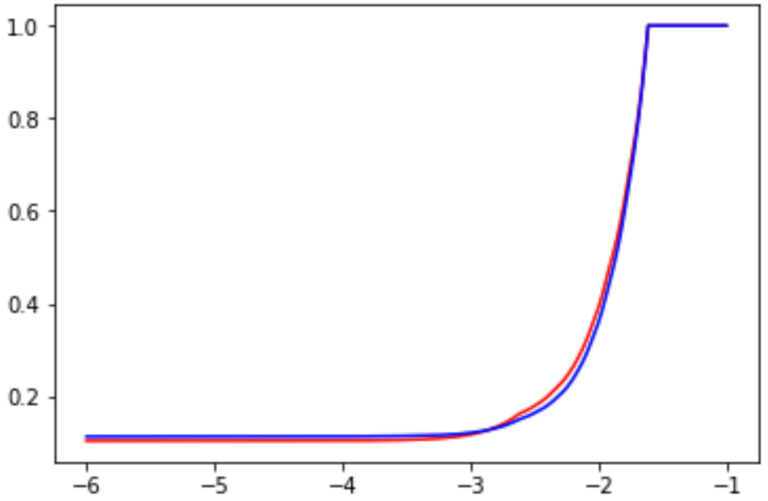


Example: house price with 16 features

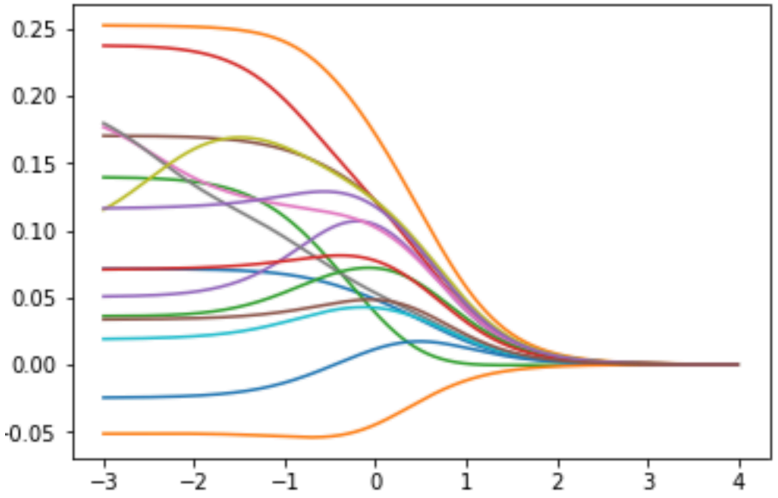
test error is red and train error is blue

error



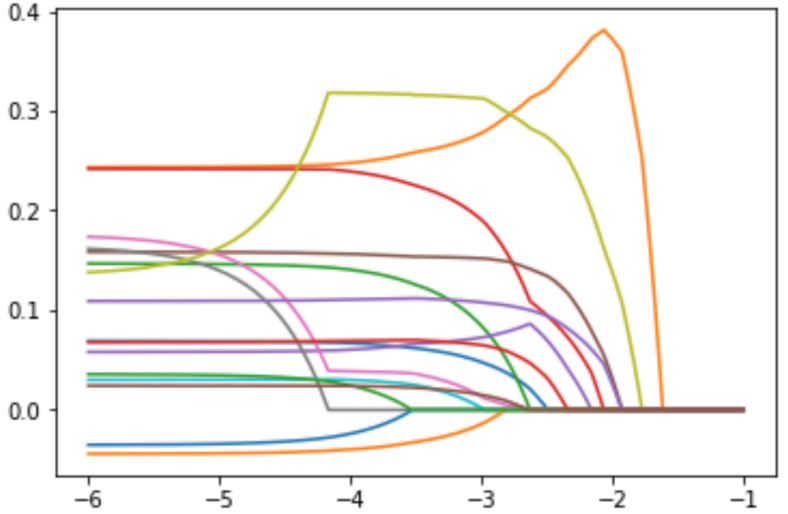


w_i's



$\log_{10} \lambda$

Ridge regression



$\log_{10} \lambda$

Lasso regression

Lasso regression naturally gives sparse features

- **feature selection** with Lasso regression
  1. choose λ based on cross validation error
  2. keep only those features with non-zero (or not-too-small) parameters in w at optimal λ
  3. **retrain** with the sparse model and $\lambda = 0$

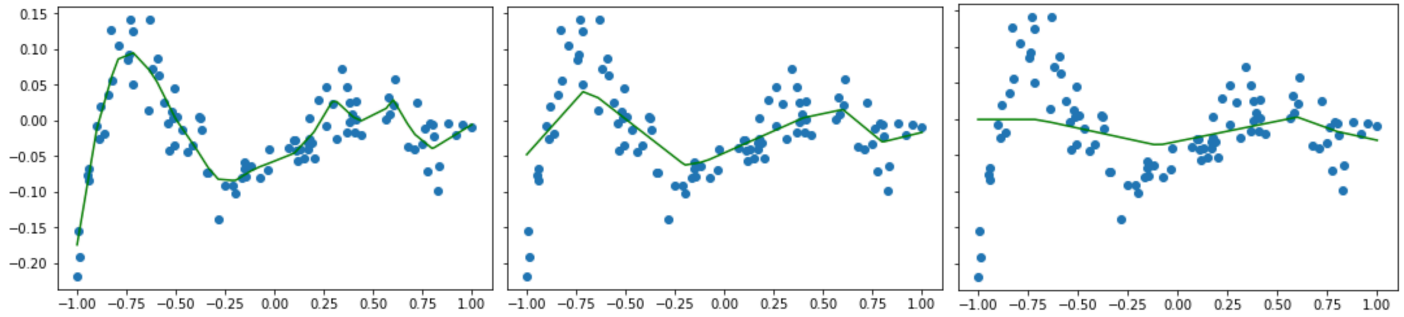
Example: piecewise-linear fit

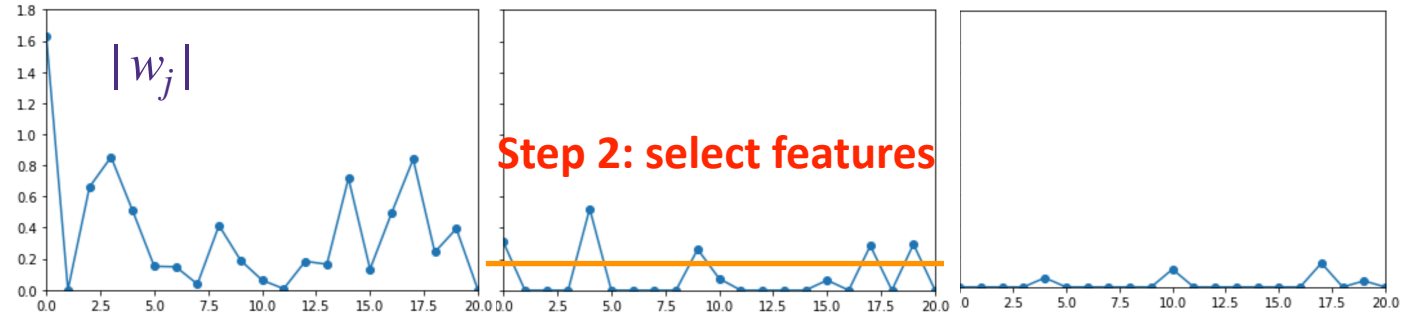
- We use Lasso on the piece-wise linear example

$$h_0(x) = 1$$
$$h_i(x) = [x + 1.1 - 0.1i]^+$$

Step 1: find optimal λ^*

$$\text{minimize}_w \quad \mathscr{L}(w) + \lambda \|w\|_1$$





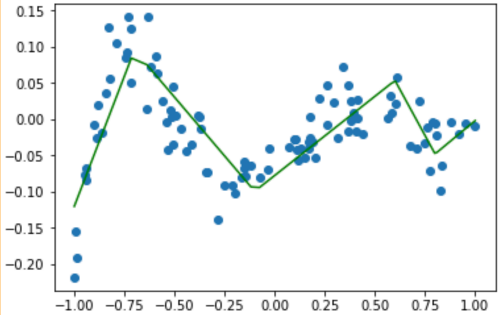
Step 2: select features

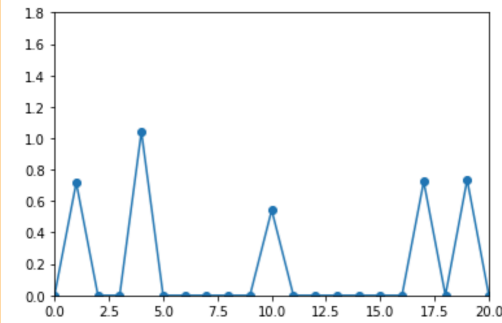
$$\lambda = 10^{-8} \qquad \lambda = 10^{-4} \qquad \lambda = 2 \times 10^{-4}$$

- de-biasing (via re-training) is critical!

Step 3: retrain

$$\text{minimize}_w \quad \mathscr{L}(w)$$





$$\lambda = 0$$

but only use selected features

Penalized Least Squares

- Regularized optimization:

$$\hat{w}_r = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda r(w)$$

$$\text{Ridge} : r(w) = ||w||_2^2$$

$$\text{Lasso} : r(w) = ||w||_1$$

- For any $\lambda^* \geq 0$ for which $\hat{w}_r$ achieves the minimum, there exists a $\mu^* \geq 0$ such that the solution of the constrained optimization, $\widehat{w}_c$, is the same as the solution of the regularized optimization , $\widehat{w}_r$, where

$$\widehat{w}_C = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 \qquad \text{subject to } r(w) \leq \mu^*$$

- so there are pairs of (λ, μ) whose optimal solution $\widehat{w}_r$ are the same for the regularizes optimization and constrained optimization

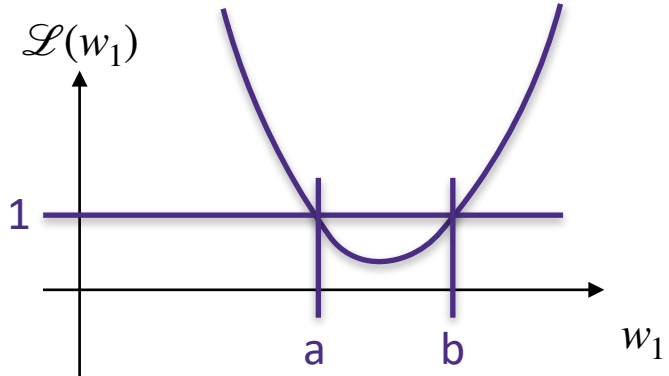
Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

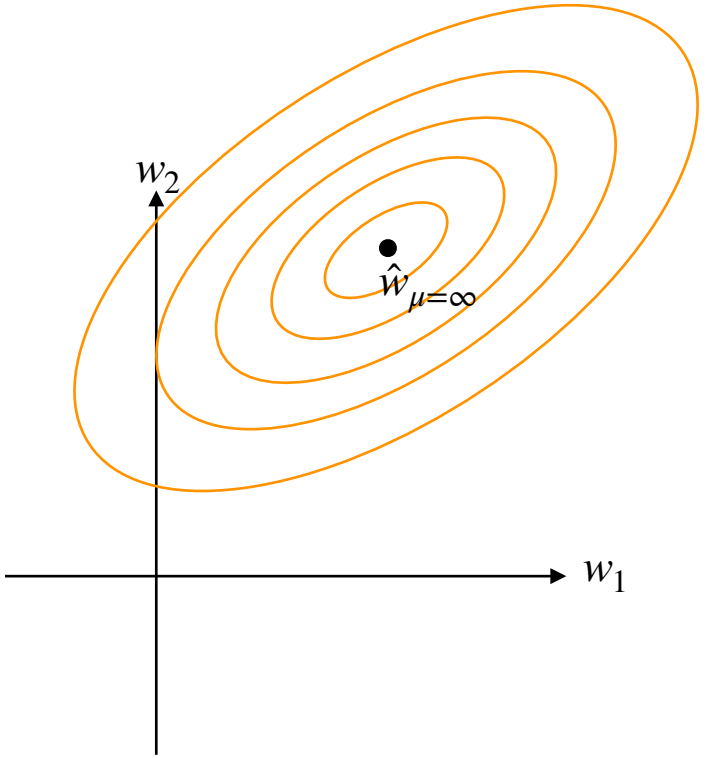
$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- the **level set** of a function $\mathscr{L}(w_1, w_2)$ is defined as the set of points (w_1, w_2) that have the same function value
- the level set of a quadratic function is an oval
- the center of the oval is the least squares solution $\hat{w}_{\mu=\infty} = \hat{w}_{\text{LS}}$

1-D example with quadratic loss



Level set of $\mathscr{L}(w_1)$ at value 1 is [a,b]



Why does Lasso give sparse solutions?

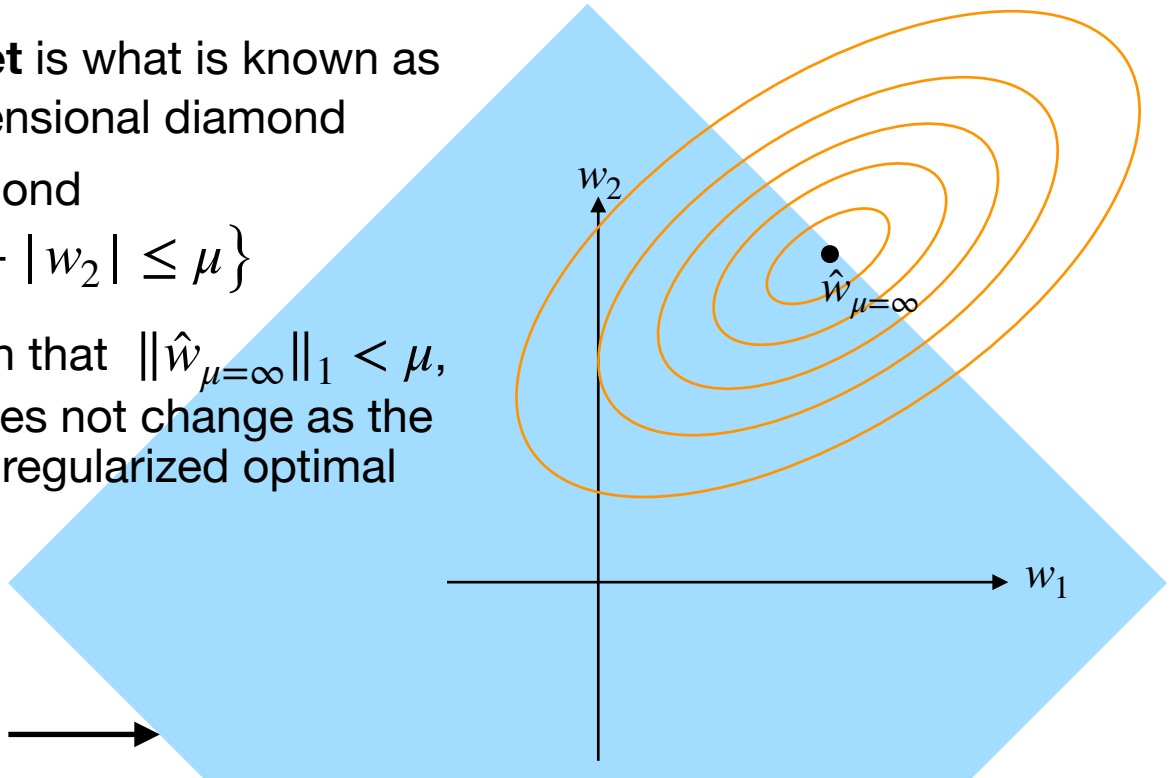
$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- as we decrease μ from infinity, the feasible set becomes smaller
- the shape of the **feasible set** is what is known as L_1 ball, which is a high dimensional diamond
- In 2-dimensions, it is a diamond

$$\{(w_1, w_2) \mid |w_1| + |w_2| \leq \mu\}$$

- when μ is large enough such that $\|\hat{w}_{\mu=\infty}\|_1 < \mu$, then the optimal solution does not change as the feasible set includes the un-regularized optimal solution



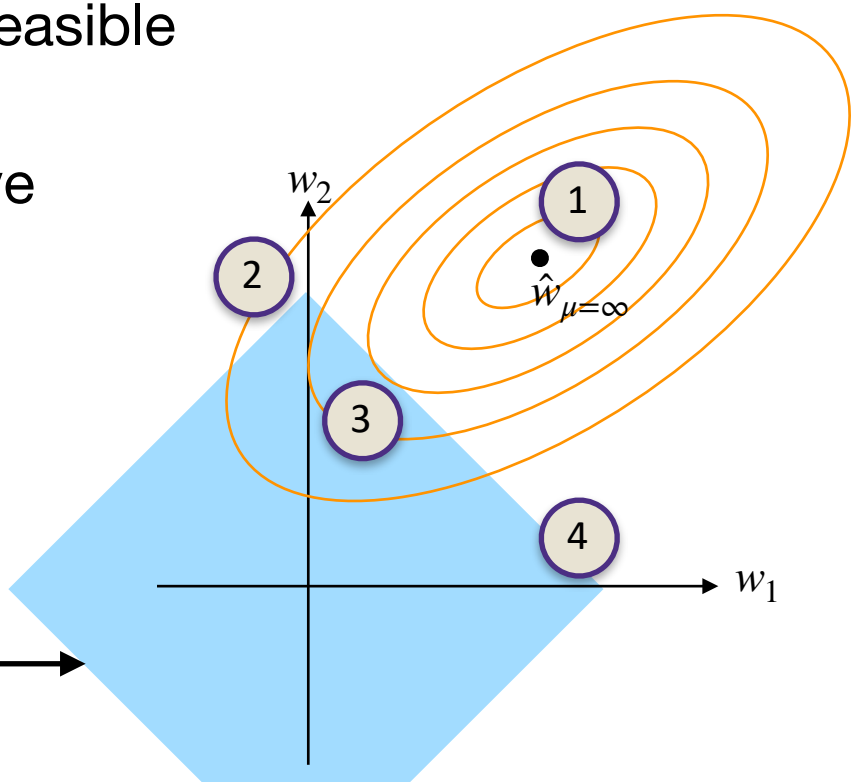
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$ ⟶

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- As μ decreases (which is equivalent to increasing regularization λ) the feasible set (blue diamond) shrinks
- The optimal solution of the above optimization is ?



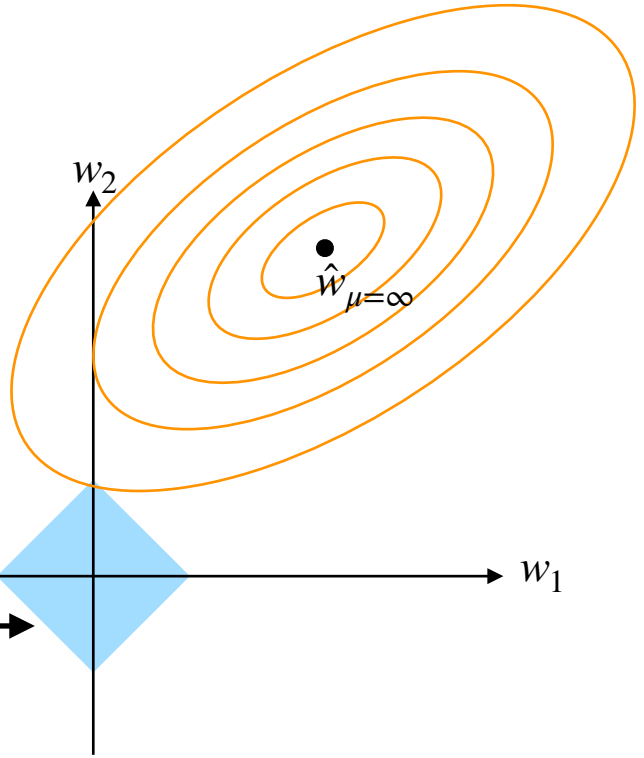
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

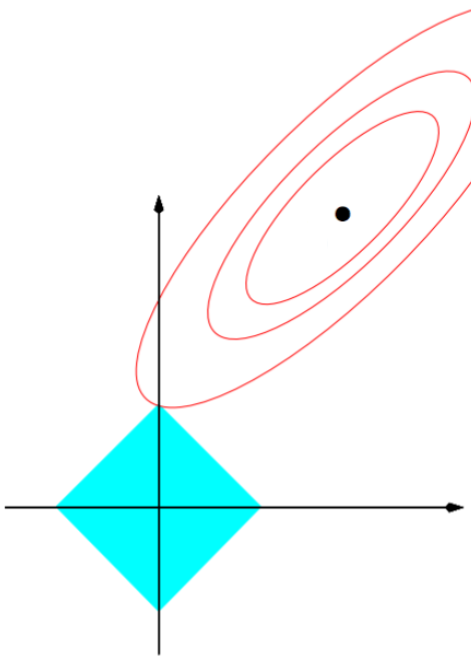
- For small enough μ, the optimal solution becomes **sparse**
- This is because the L_1-ball is "pointy",i.e., has sharp edges aligned with the axes

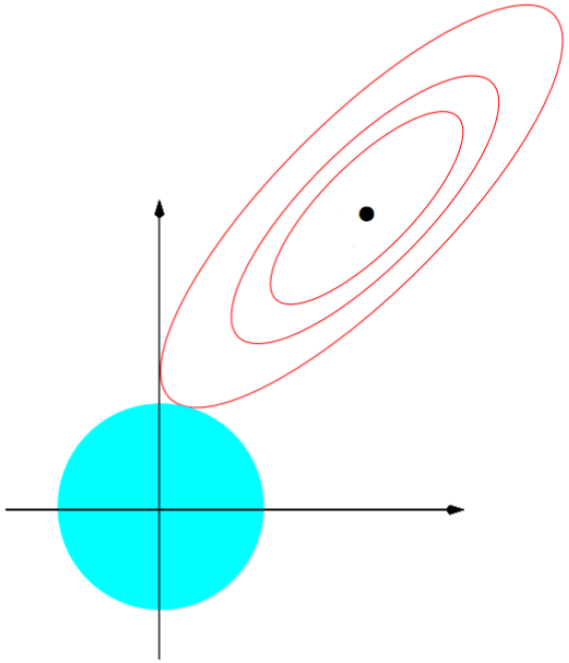


feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Penalized Least Squares

- Lasso regression finds sparse solutions, as L_1-ball is "pointy"
- Ridge regression finds dense solutions, as L_2-ball is "smooth"





$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_2^2 \leq \mu$$

Ridge vs. Lasso

- **Ridge**
  - Very fast:
    - Closed form solution if used with linear models
    - Even with non-linear and complex loss, optimization is fast for squared ℓ_2 regularization (to be taught later)
  - Gives regularized parameters that avoid overfitting

- **Lasso**
  - Slower than Ridge:
    - No closed form!
    - A non-smooth optimization which is slower
      - (to be taught later)
  - Gives sparse parameters

Questions?

Lecture 10: Convexity

- When is an optimization (or learning) easy/fast to solve?

Recap: Ridge vs. Lasso

- **Ridge**

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$$

  - Very fast:
    - Closed form solution if used with linear models
    - Even with other loss functions, optimization is fast for squared ℓ_2 regularization, because $\|w\|_2^2$ is **convex and smooth**

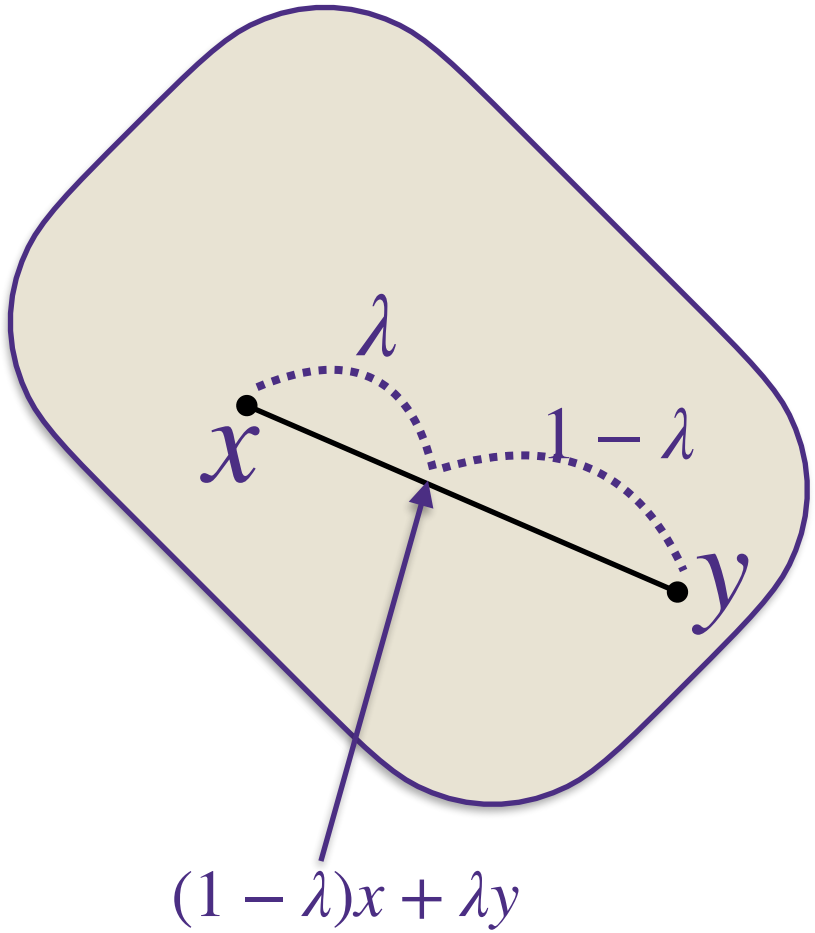
- **Lasso**

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_1$$

  - Slower than Ridge:
    - Requires iterative optimization algorithm like sub-gradient descent
    - In particular, it is slower because $\|w\|_1$ is **convex but non-smooth**

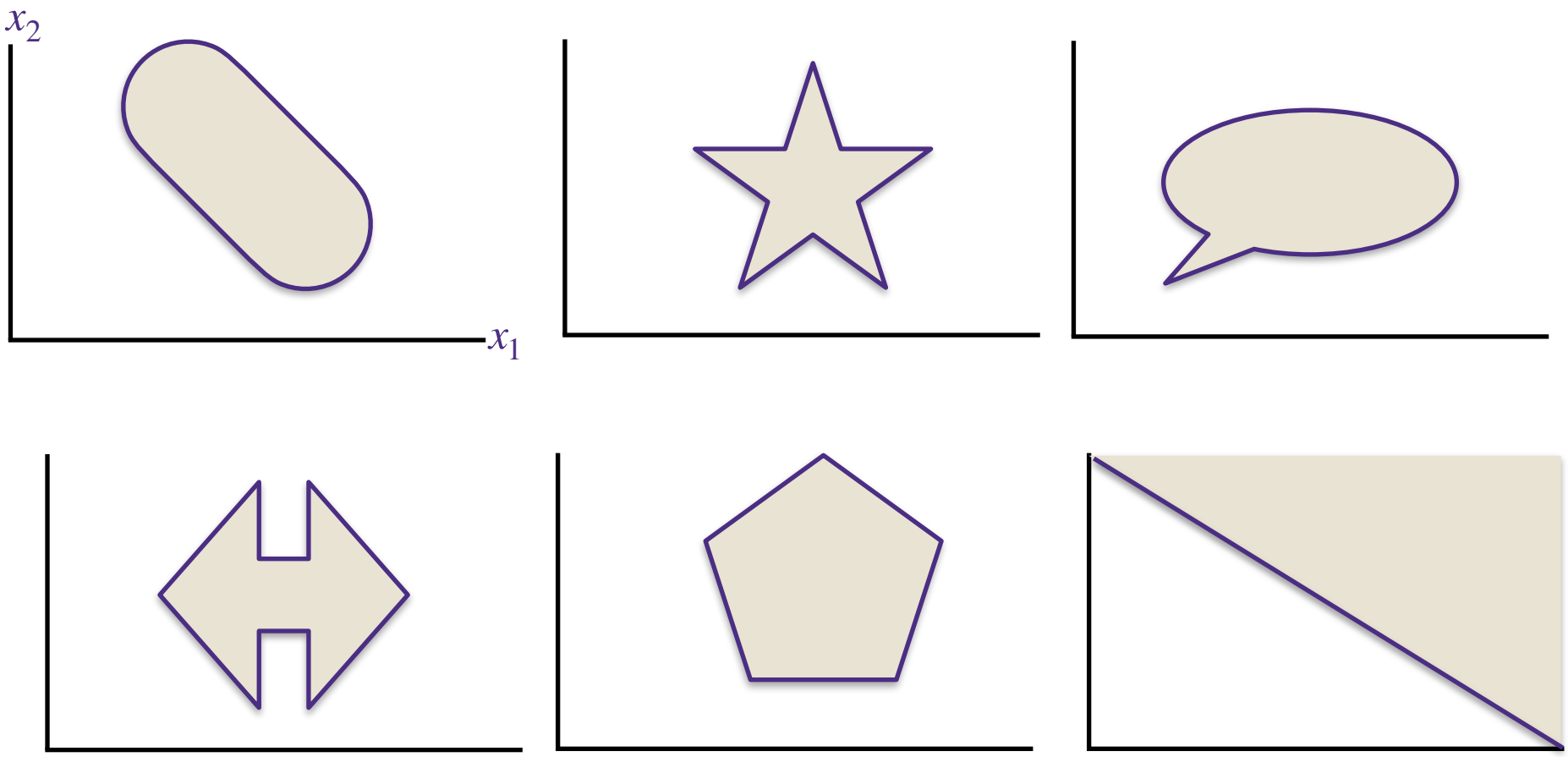
What is a convex set?

A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$



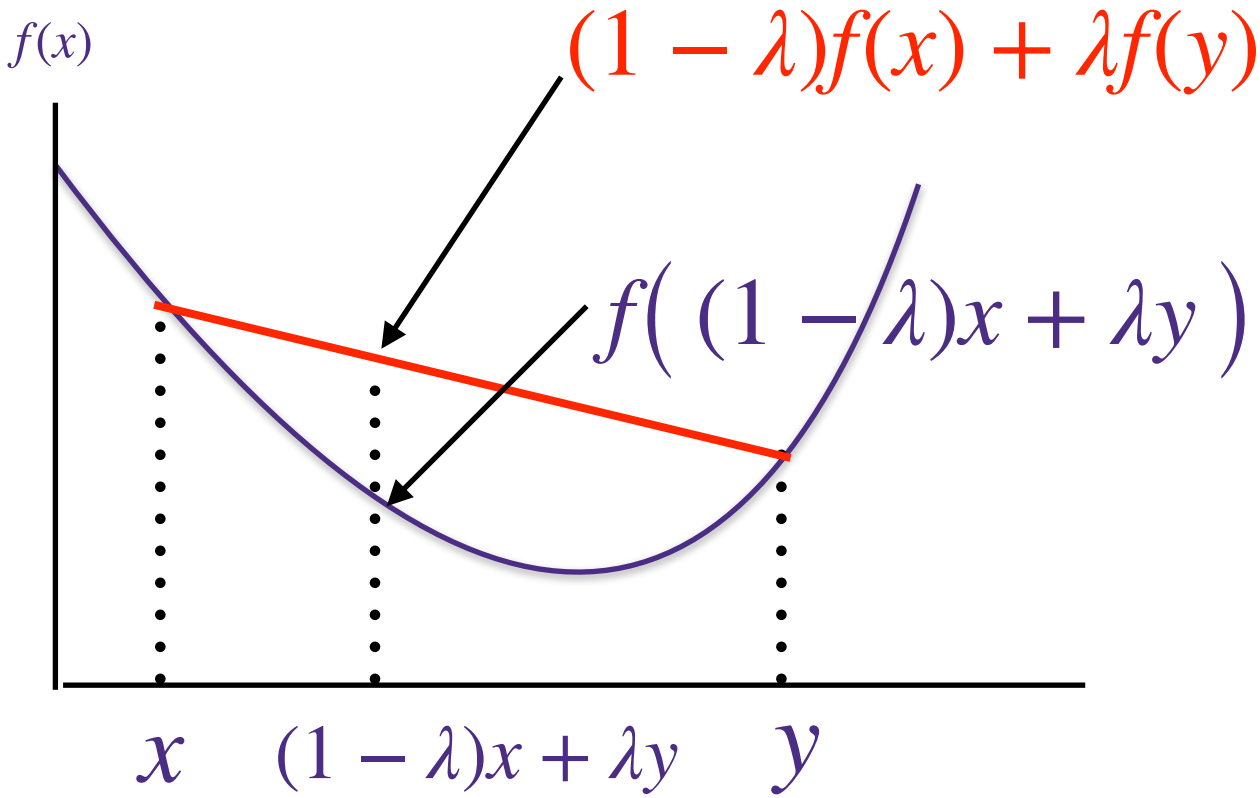
What is a convex set?

A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$



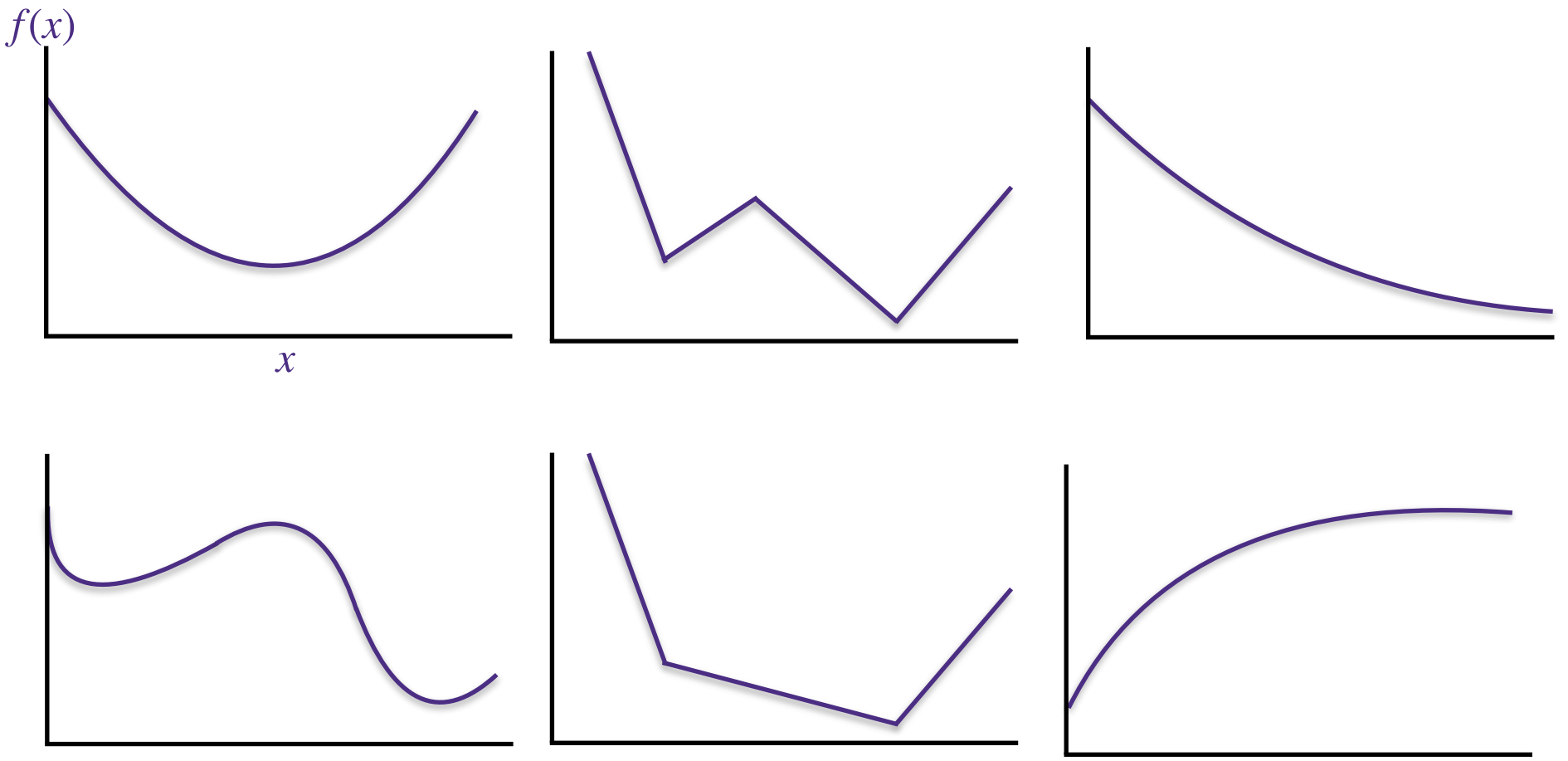
What is a convex function?

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$



What is a convex function?

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$



Convex functions and convex sets?

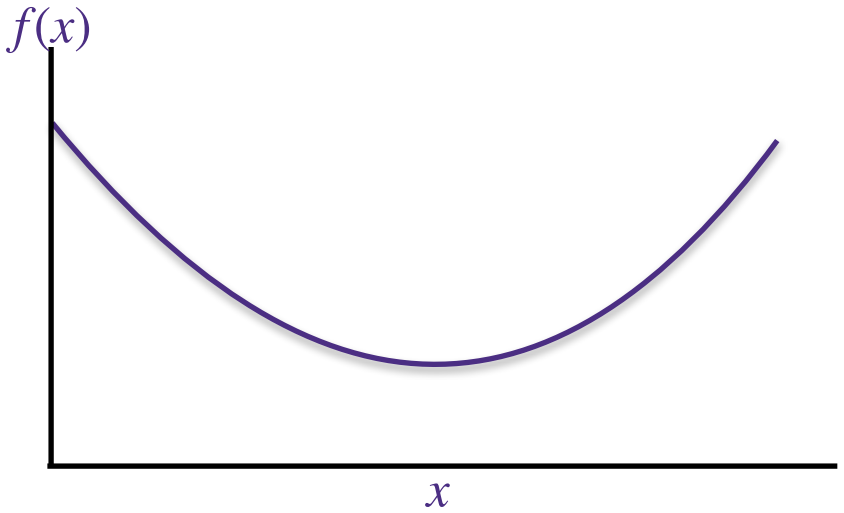
A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$

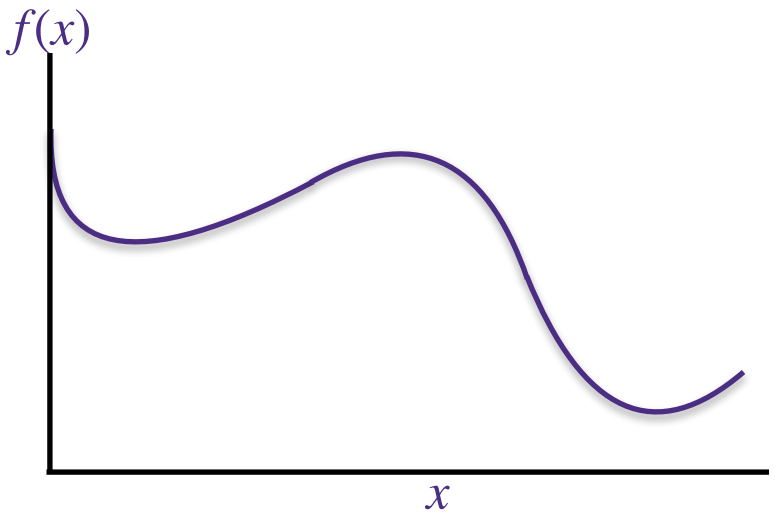
A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the set $\{(x,t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

Graph of f is defined as $\{(x,t) : f(x) = t\}$

Epigraph of f is defined as $\{(x,t) : f(x) \leq t\}$



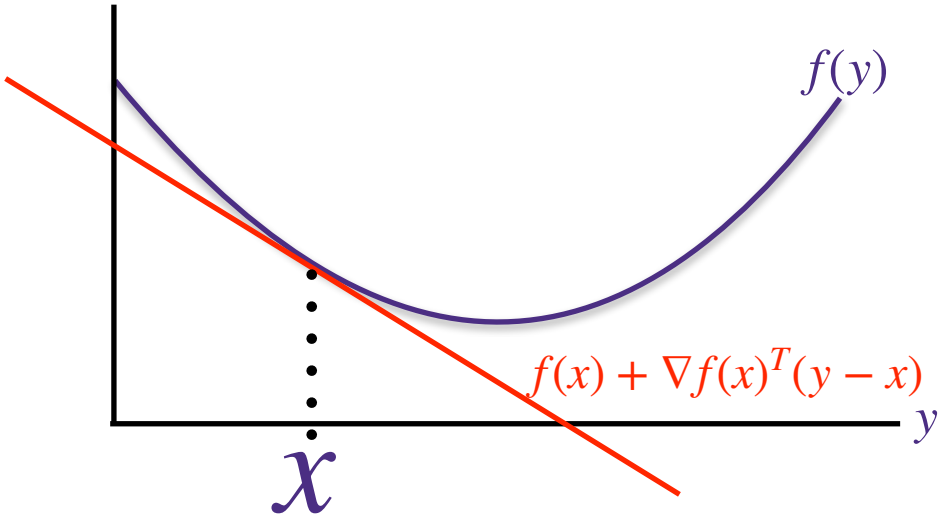


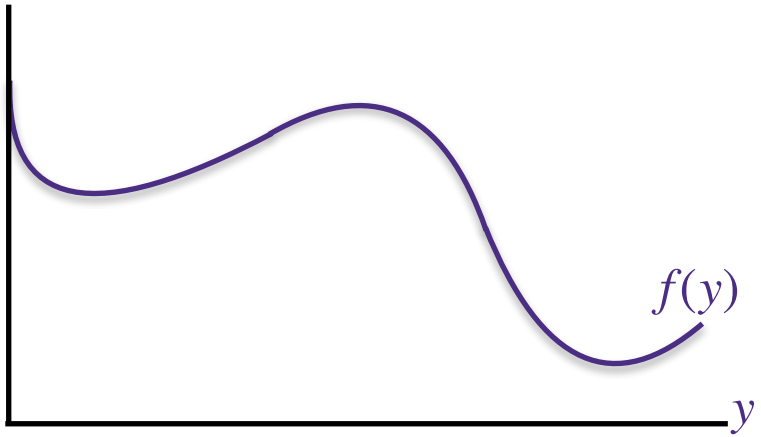
More definitions of convexity

A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the set $\{(x,t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

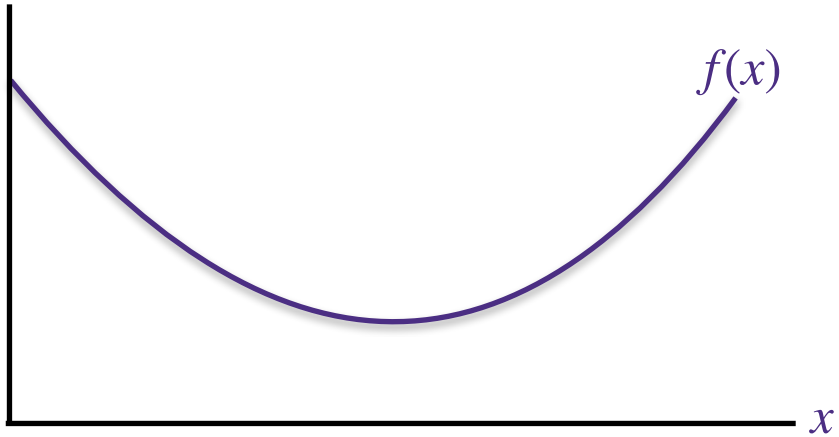
A function $f : \mathbb{R}^d \to \mathbb{R}$ that is differentiable everywhere is convex if $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in dom(f)$

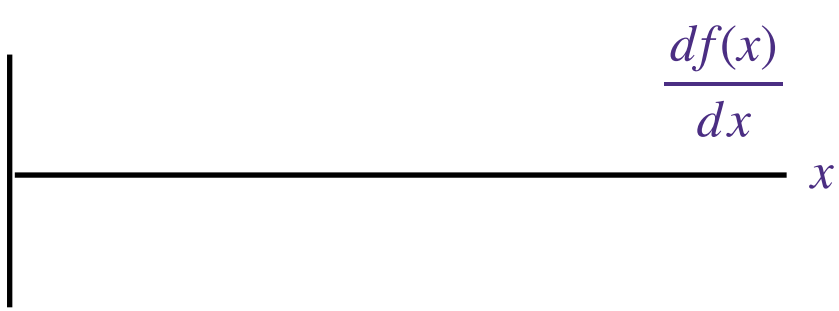


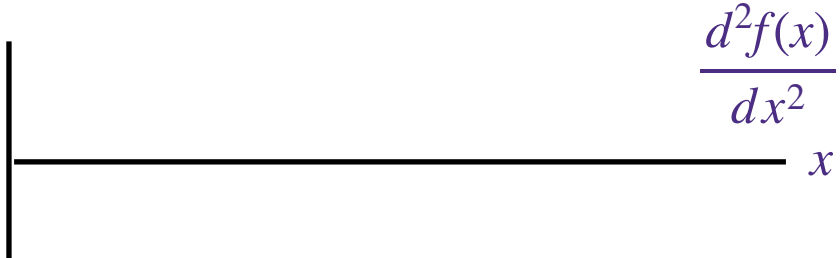


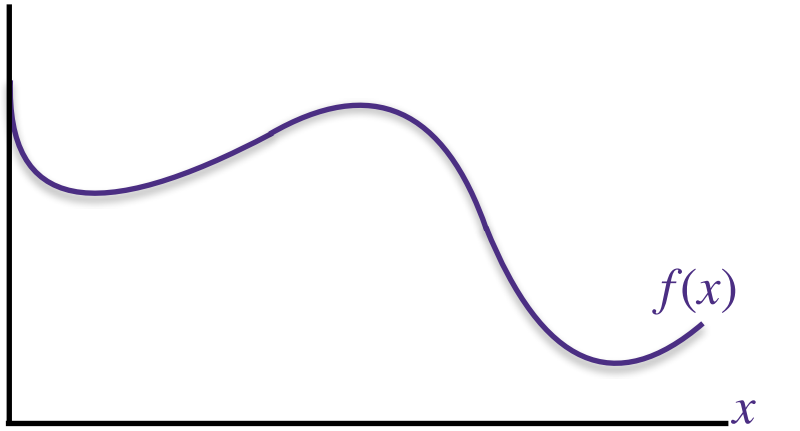
More definitions of convexity

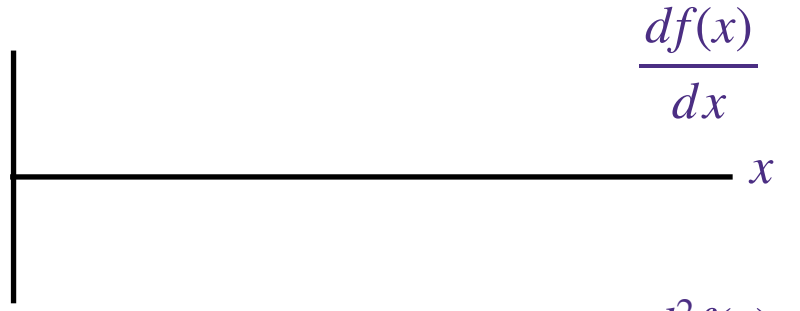
A function $f : \mathbb{R}^d \to \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in dom(f)$

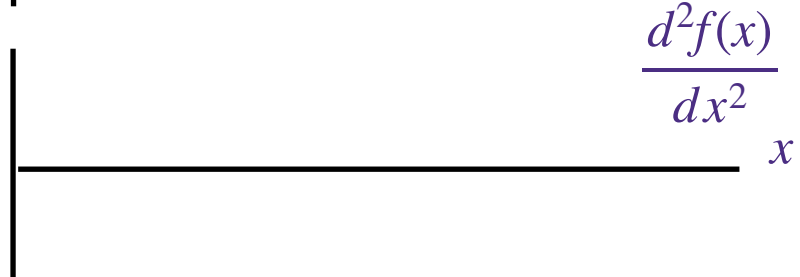












More definitions of convexity

A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the set $\{(x,t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

A function $f : \mathbb{R}^d \to \mathbb{R}$ that is differentiable everywhere is convex if $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in dom(f)$

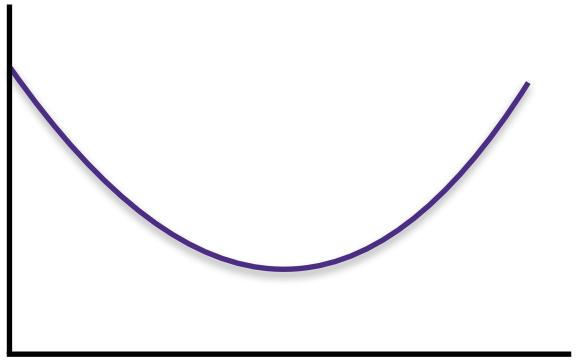
A function $f : \mathbb{R}^d \to \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in dom(f)$

Why do we care about convexity?

Convex functions

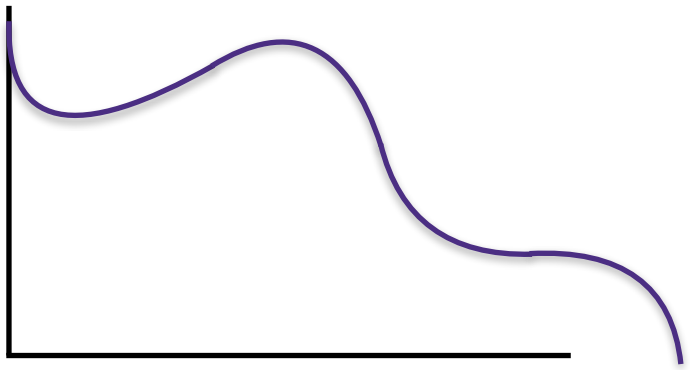
- All local minima are global minima
- Efficient to optimize (e.g., gradient descent)

Convex Function



We only need to find a point with $\nabla f(x) = 0$, which for convex functions implies that it is a local minima and a global minima

Non-convex Function



For non-convex functions, a stationary point with $\nabla f(x) = 0$ could be a local minima, a local maxima, or a saddle point

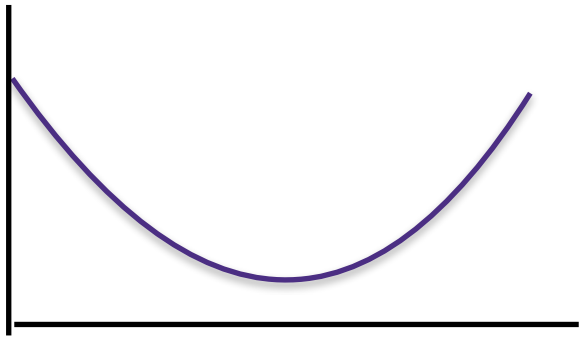
Gradient Descent on $\min_w f(w)$

Initialize: $w_0 = 0$

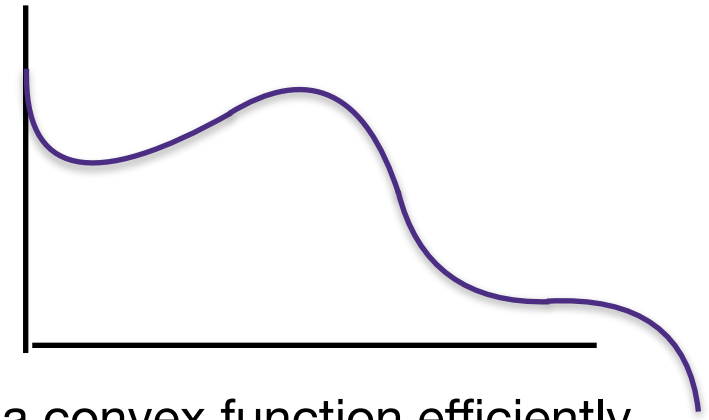
for $t = 1, 2, \ldots$

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

Convex Function



Non-convex Function



- Strength: Can find global minima of a convex function efficiently
- Weakness: Can only be applied to smooth functions
  - i.e., functions that is differentiable everywhere,
  - otherwise $\nabla f(x)$ is not defined and gradient descent cannot be applied

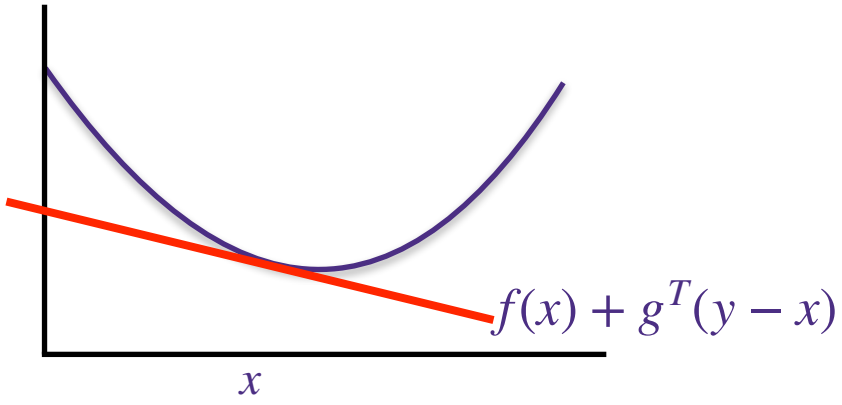
Sub-Gradient

Definition: a function is **non-smooth** if it is not differentiable everywhere

Definition: a vector $g \in \mathbb{R}^d$ is a **sub-gradient** at x if it satisfies

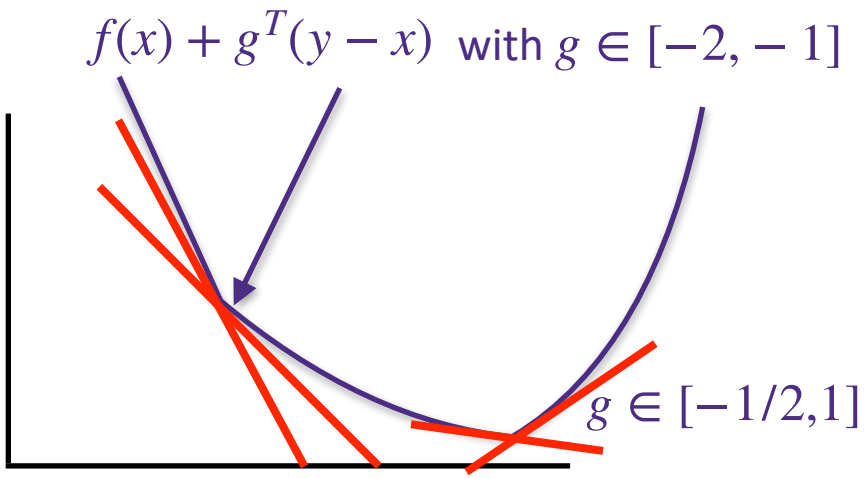
$$f(y) \geq f(x) + g^T(y - x) \text{ for all } y \in \mathbb{R}^d$$

Smooth Convex Function



- for smooth convex functions,
  - gradient is the unique sub-gradient, and
  - the global minimum is achieved at points where gradient is zero

Non-smooth Convex Function



- for non-smooth convex functions,
  - the minimum is achieved at points where sub-gradient set includes the zero vector

Sub-Gradient Descent for non-smooth functions

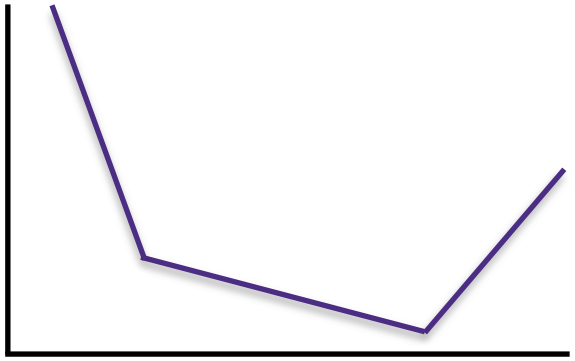
Initialize: $w_0 = 0$

for $t = 1, 2, \ldots$

Find any g_t such that $f(y) \geq f(w_t) + g_t^\top (y - w_t)$

$$w_{t+1} \leftarrow w_t - \eta_t g_t$$

Convex Function



Non-convex Function

- Strength: finds global minima for **non-smooth convex functions**
- Weakness: it is slower than gradient descent on convex smooth functions, because the gradient do not get smaller near the global minima
  - Instead of last iterate w_t, we use the best one we saw in all iterates
  - The stepsize needs to decrease with t

Coordinate descent

Initialize: $w_0 = 0$

for $t = 1, 2, \ldots$

Let $i_t = t \ \% \ d$

$$w_{t+1}[i_t] \leftarrow w_t[i_t] - \eta_t \frac{\partial f(w_t)}{\partial w[i_t]}$$

Optimization

- You can always run gradient descent whether f is convex or not. But you only have guarantees if f is convex
- Many bells and whistles can be added onto gradient descent such as momentum and dimension-specific step-sizes (Nesterov, Adagrad, ADAM, etc.)

Questions?