Announcements

- Pset1 out, due in a week.
  - Submit on Gradescope
  - Questions:
    - Look at previous posts. If not answered, then post them on piazza.

- Quiz-1 : timed for 1 hour.
  - Very similar to the slido questions we discuss in the class.

- It is in your best interest to answer them sincerely.

Last time

- Bayesian Modeling
- Practical notes on how to train a model.

Topics today

- Interesting questions after class
- Loss functions
- The continued saga of overfitting
  - Regularizers
  - Bias and variance
- SVM Intuition and Formulation

Topics today

- **Interesting questions after class**
- Loss functions
- The continued saga of overfitting
  - Regularizers
  - Bias and variance
- SVM Intuition and Formulation

What is the run time complexity of computing MLE?

MLE = $\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(u^{(i)}|\theta)$

- Highly dependent on:
  - prior
    - number of parameters to estimate (k)
    - the model complexity
  - number of datapoints (n)

Topics today

- Interesting questions after class
- **Loss functions**
- The continued saga of overfitting
  - Regularizers
  - Bias and variance
- SVM Intuition and Formulation

Training Setup

Target Function: $\gamma_i = a^T x_i + b$

Model Parameters

Training Error: $$\frac{1}{N}\sum_{i=1}^{N} C(y_i, \gamma_i)$$

where γ_i is the predicted label

also called loss function, cost function, objective function

Loss functions

- Loss functions are a way to quantify when your model is doing well.

- Computed purely based on the data

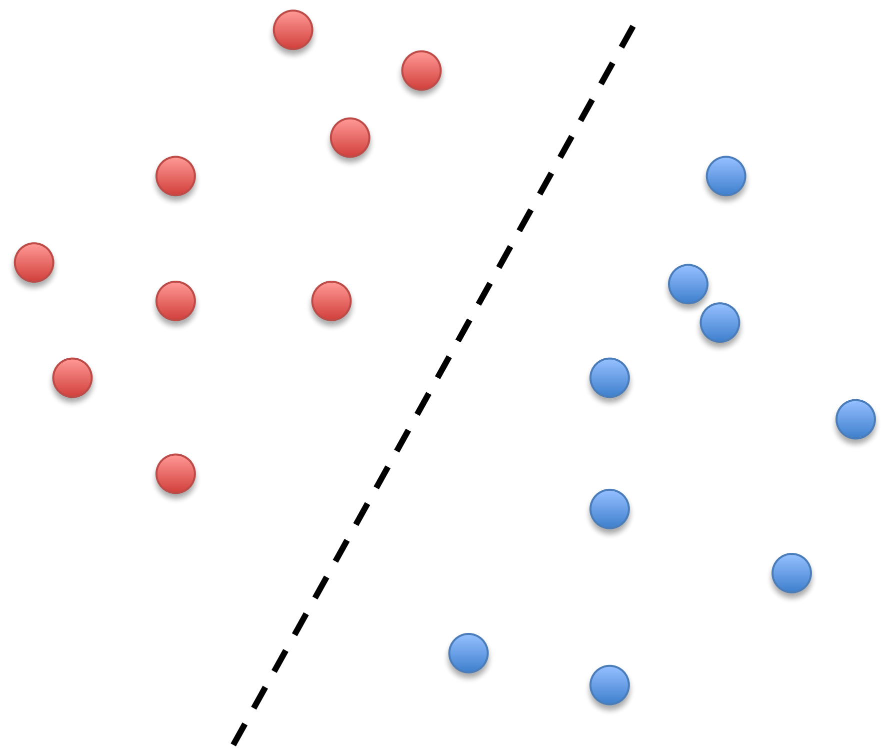
- Good model = low loss
- Bad model = high loss.

Do not edit

How to change the design

What are some good choices for a loss function?

Consider a linear classifier



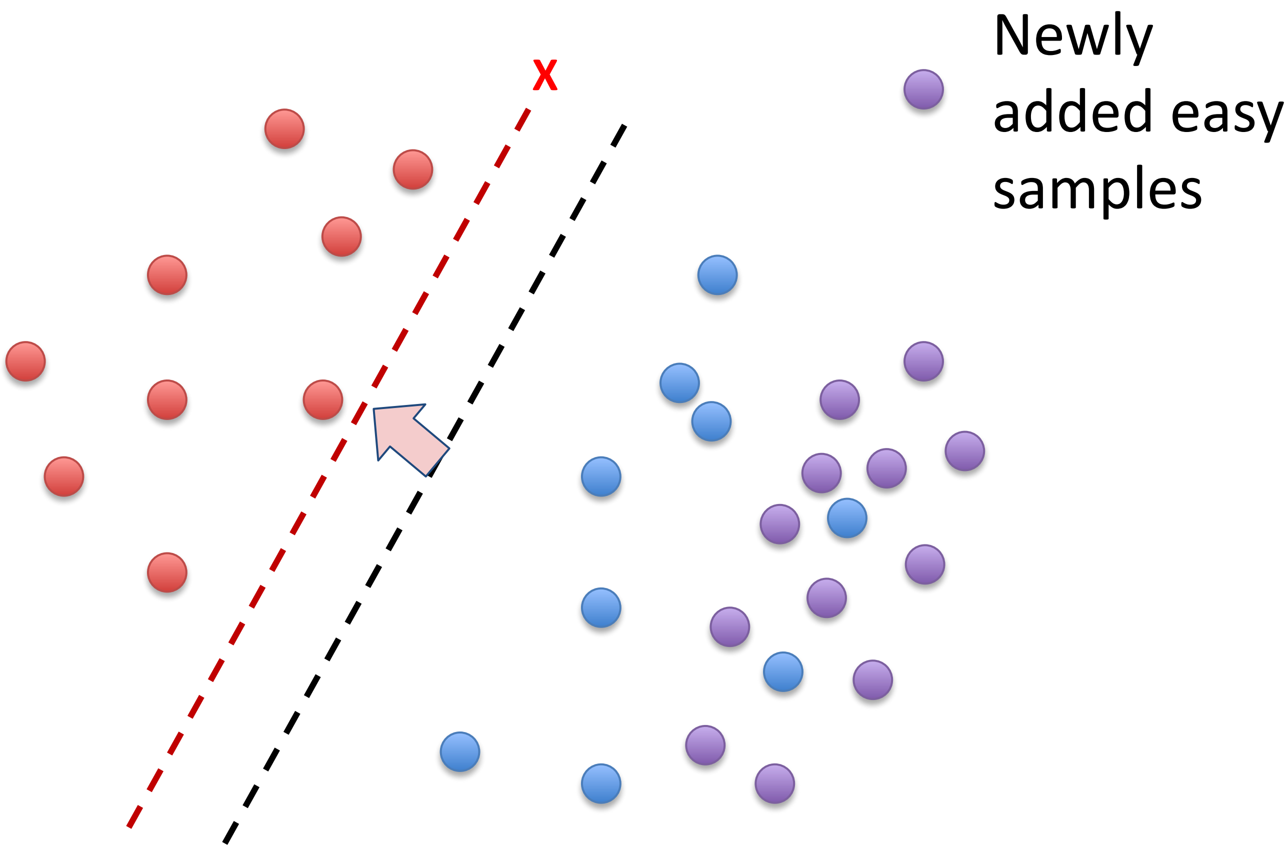
What is a good cost function C?

$$\frac{1}{N}\sum_{i=1}^{N} C(y_i, \gamma_i)$$

- **Misclassification count:** Count of the number of mis-predictions: treats all mistakes and correct predictions equally.

- **Mean Squared Error:** Penalizes errors more heavily than correct predictions.

What happens when we have a lot of easy samples?



Goal: Do not shift the decision boundary because of easy samples.

Topics today

- Interesting questions after class
- Loss functions
- **The continued saga of over-fitting**
  - **Regularizers**
  - Bias and variance
- SVM Intuition and Formulation

Training Setup

Target Function: $\gamma_i = a^T x_i + b$

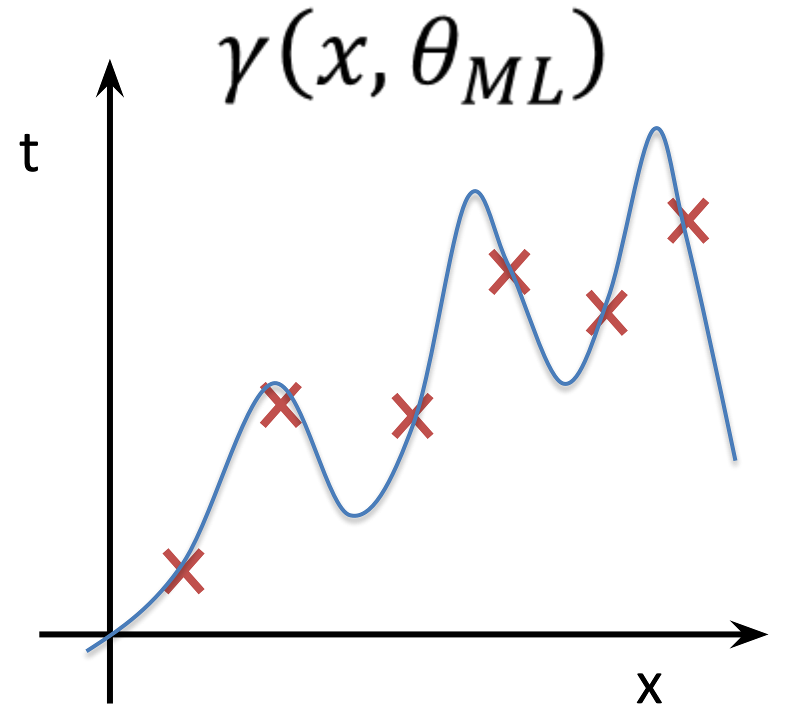
Model Parameters

Training Error: $$\frac{1}{N}\sum_{i=1}^{N} C(y_i, \gamma_i)$$

where γ_i is the predicted label

Recall: Overfitting

- Learning a model that performs too well on the training data.
- May not generalize to unseen test samples.



How can you detect overfitting?

How can you detect overfitting?

Multiple Choice Poll 78 votes 78 participants

Observe validation loss - 13 votes — 17%

Observe test loss - 8 votes — 10%

Both - 58 votes — 74%

None - 2 votes — 3%

slido

How to avoid it? Regularization

Target Function: $\gamma_i = a^T x_i + b$

Model Parameters

Training Error: $$\frac{1}{N}\sum_{i=1}^{N} C(y_i, \gamma_i)$$

where γ_i is the predicted label

- **Regularizer:** A penalty term included in the training objective to discourage making large errors on new data

Regularization

Target Function: $\gamma_i = a^T x_i + b$

Model Parameters

Training Error: $$\frac{1}{N}\sum_{i=1}^{N} C(y_i, \gamma_i)$$

where γ_i is the predicted label

- **Regularizer:** A penalty term included in the training objective to discourage making large errors on new data

Regularization Term: $\lambda\left(\frac{a^T a}{2}\right)$

Overall training objective

λ = regularization strength (hyperparameter)

$$\text{Loss} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} C(y_i, \gamma_i)}_{} + \underbrace{\lambda\left(\frac{a^T a}{2}\right)}_{}$$

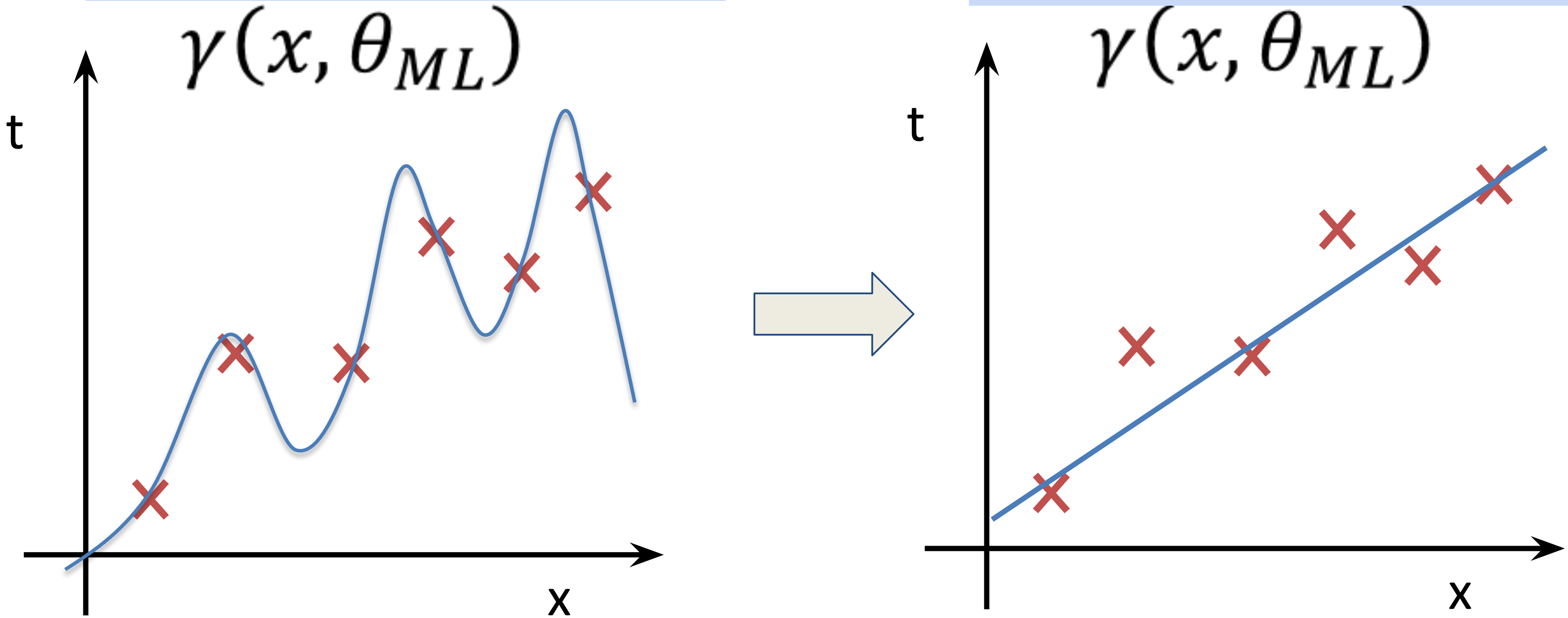
Data loss: Model predictions should align with ground truth

Regularizer: Prevent the model from doing too well.

Slide motivation: Justin Johnson

What does a regularizer do?

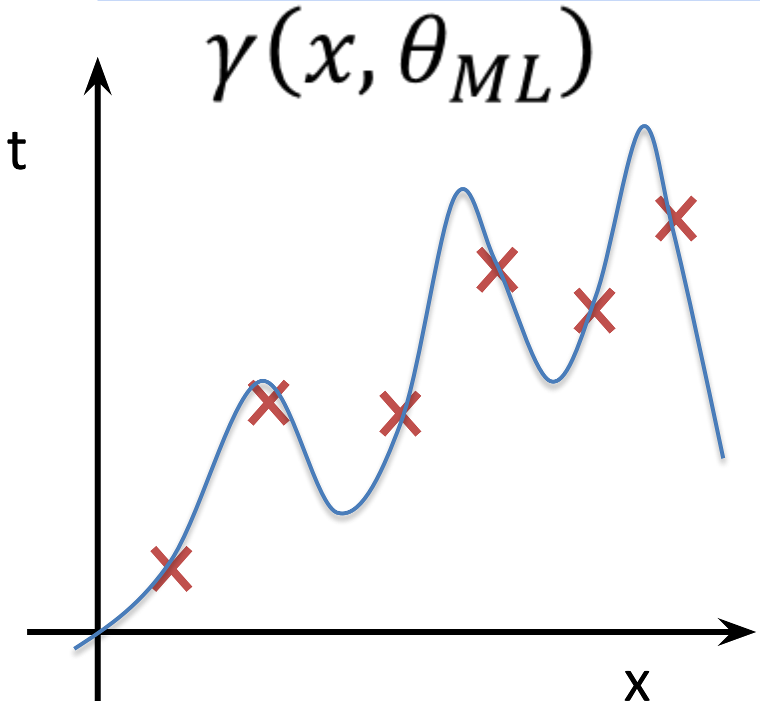
Before regularizing | After regularizing



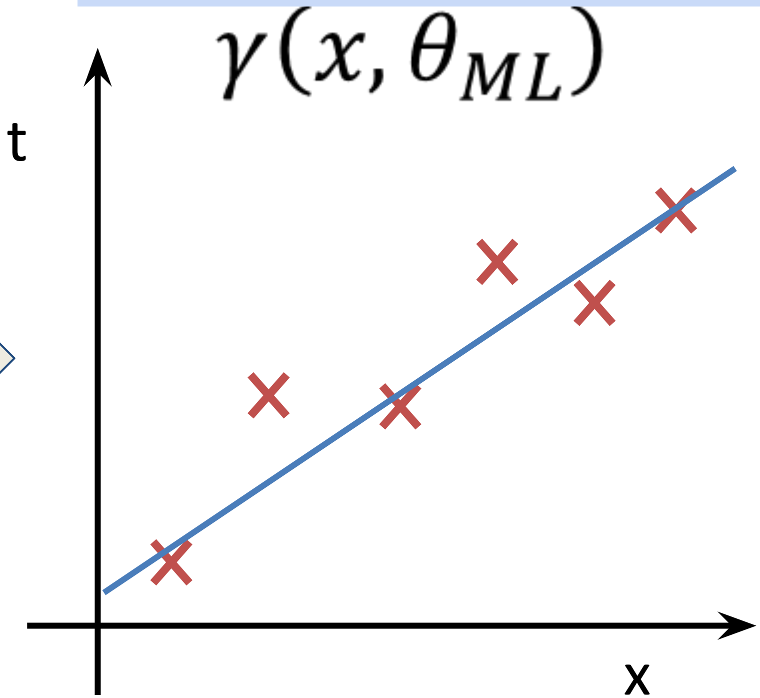
Why does this work? 

What does a regularizer do?

Before regularizing



After regularizing



Simplifies the model by dampening the coefficients of a few parameters.

Why not choose a simple model instead of adding a regularizer? (choose all that apply)

Why not choose a simple model instead of adding a regularizer? (choose all that apply)

Multiple Choice Poll · 77 votes · 77 participants

A simple model may not always be representative of the training data distribution - 58 votes

75%

A simple model may lead to a high data loss which is undesirable - 40 votes

52%

True. Choosing a simple model makes more sense than adding regularizer. - 4 votes

5%

Regularizers offer the ability to choose complex model but still avoid overfitting - 65 votes

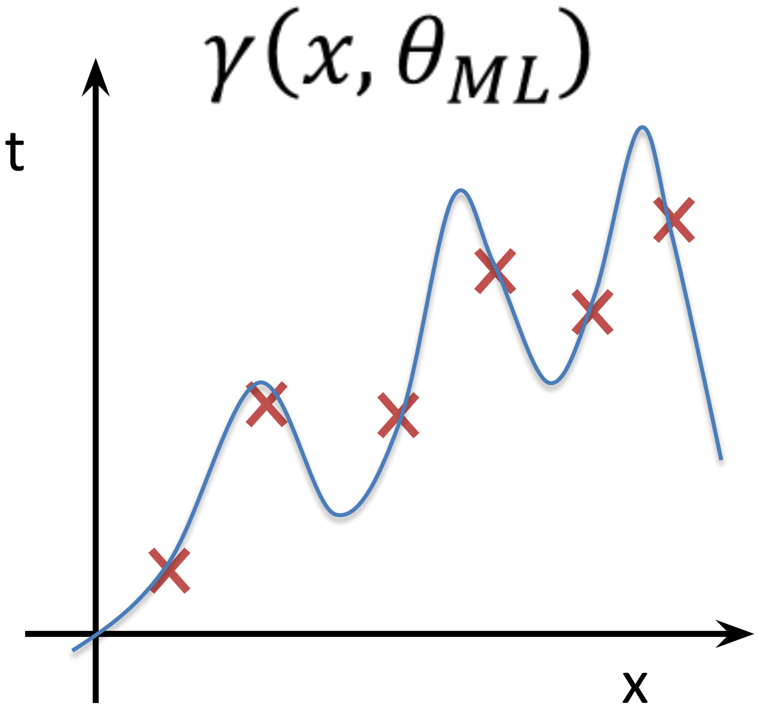
84%

slido

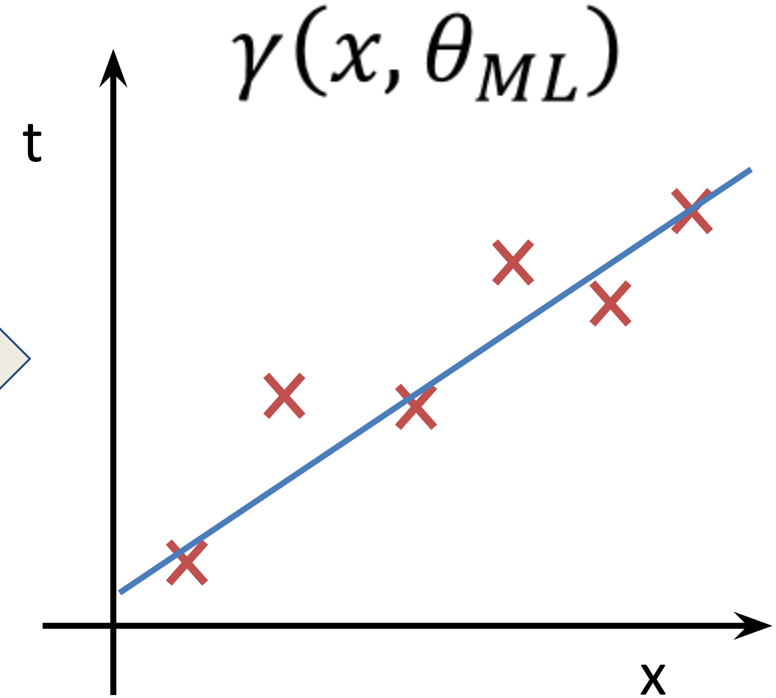
Topics today

- Interesting questions after class
- Loss functions
- **The continued saga of over-fitting**
  - Regularizers
  - **Bias and variance**
- SVM Intuition and Formulation

Bias and Variance



Low training error



High training error

Bias: Inability of the model to capture the true relationship between data and labels

What does it mean to have low bias?

What does it mean to have low bias?

Quiz question 73 answers 73 participants

Model has high training error - 20 answers

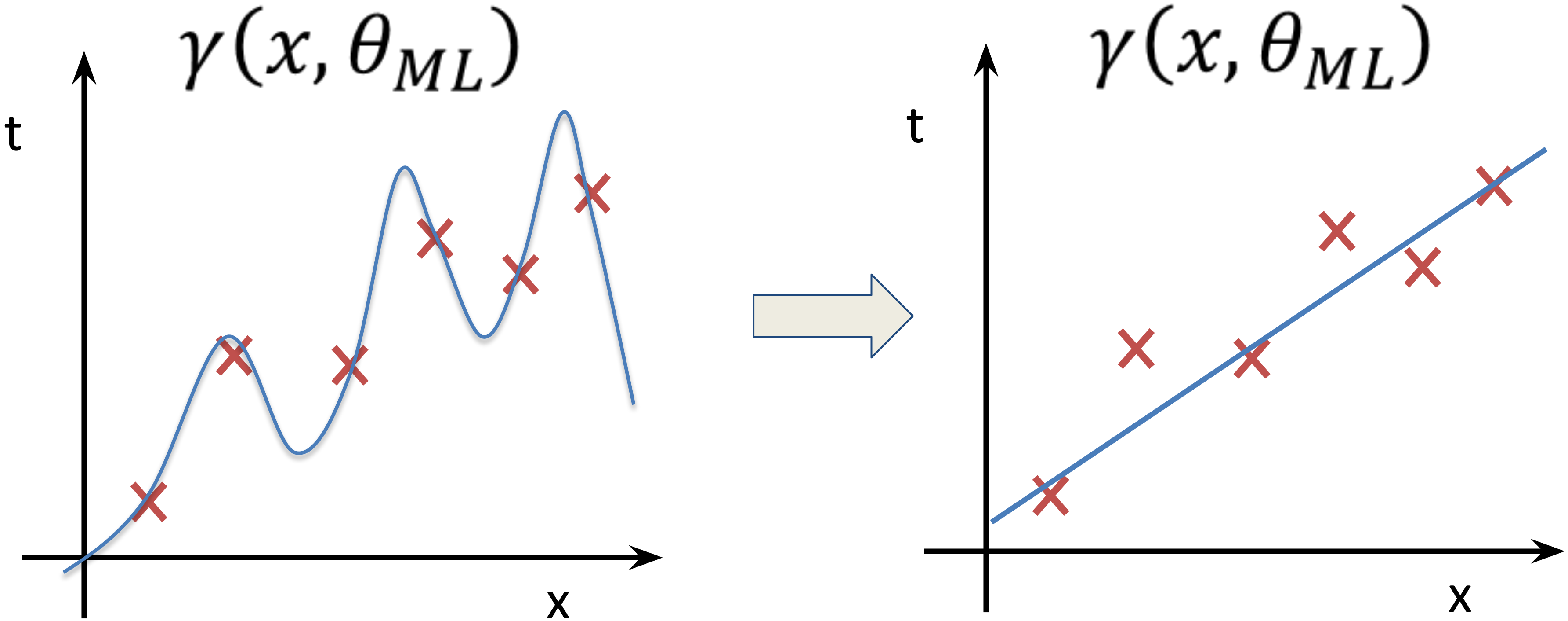
27%

Model has low training error - 53 answers

73%

slido

Bias: Inability of the model to capture the true relationship between data and labels



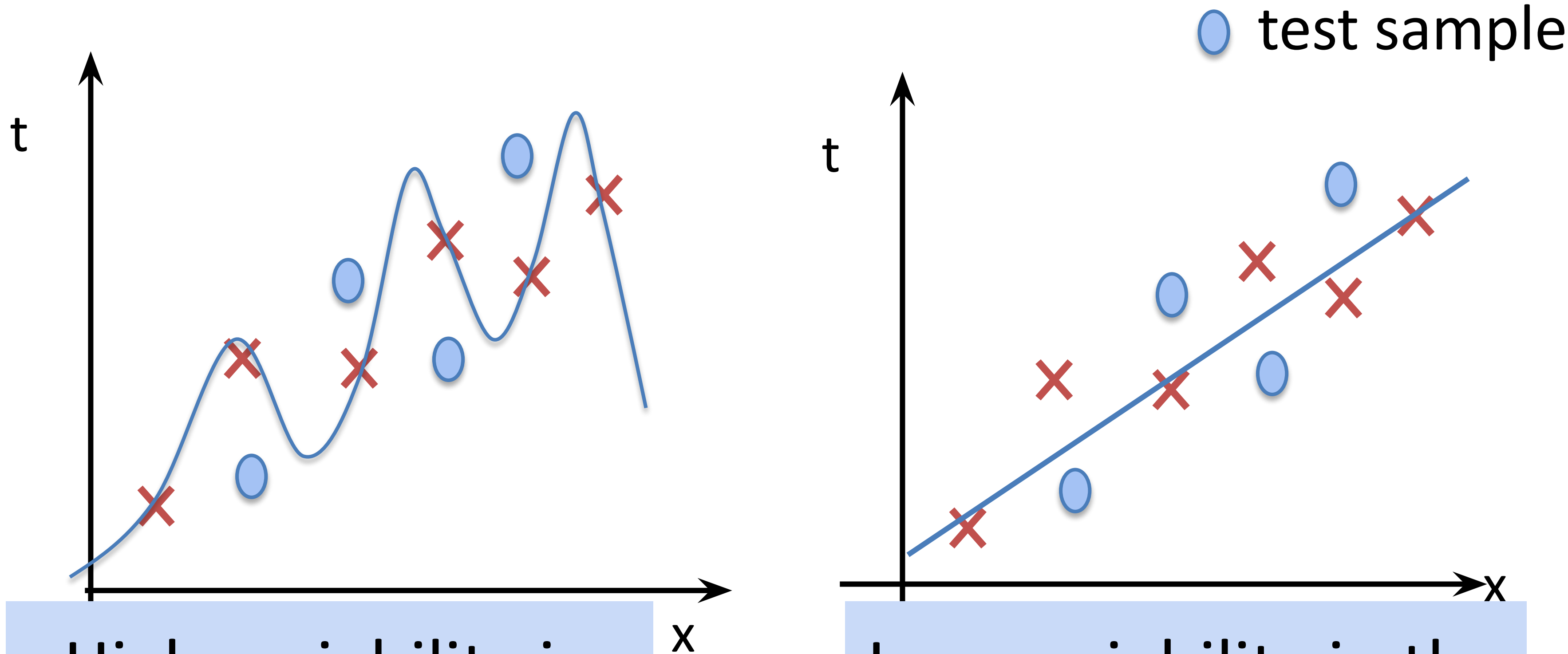
Low training error

High training error

Low bias

High bias

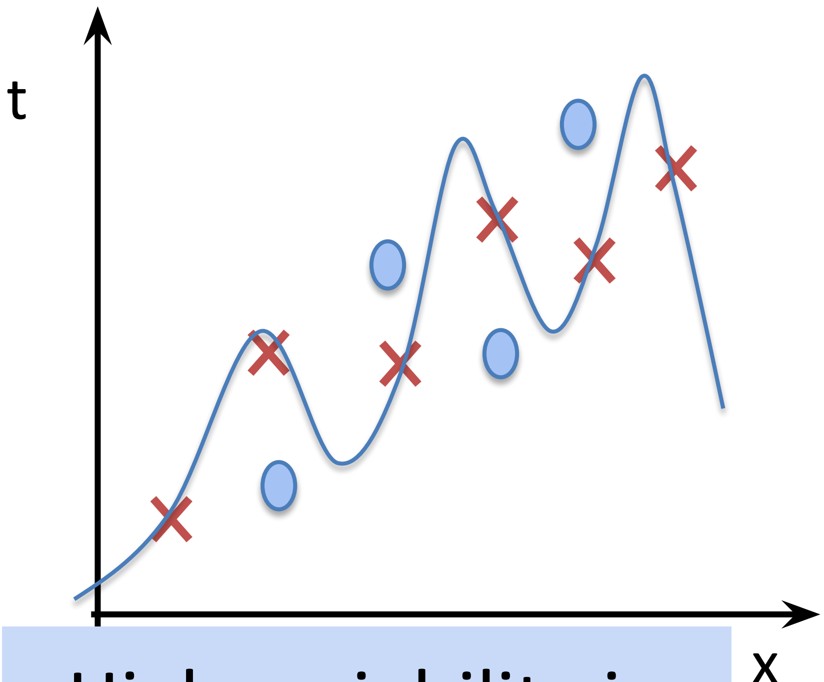
Test data

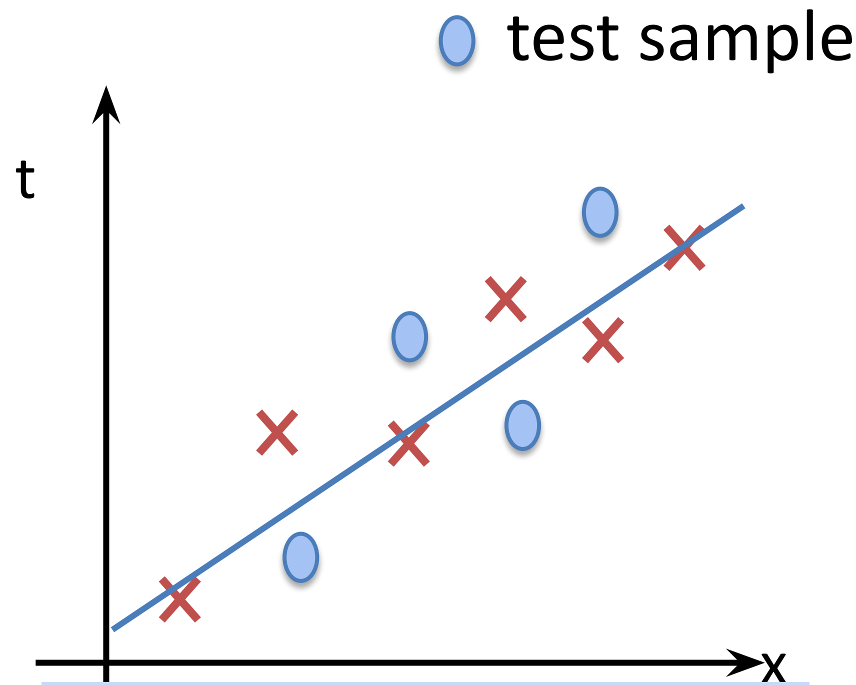


High variability in loss on test data

Low variability in the loss on test data

Test data



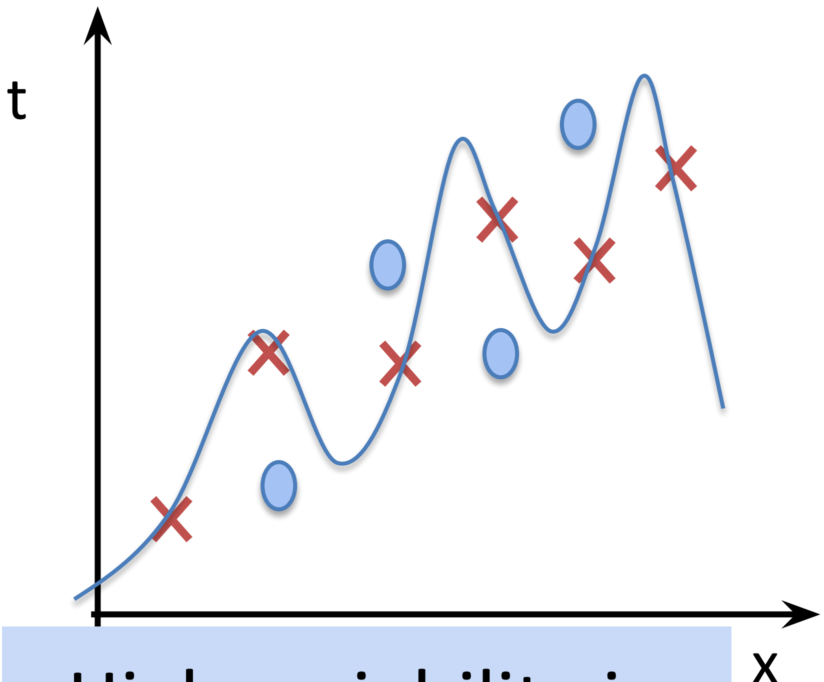


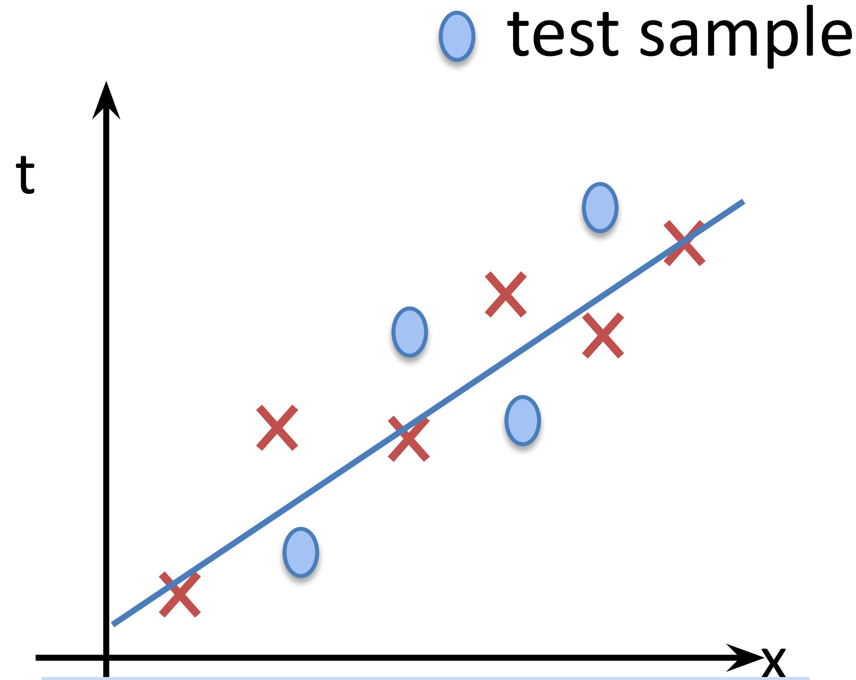
High variability in loss on test data

Low variability in the loss on test data

Unreliable on how it performs on unseen test sets

Test data



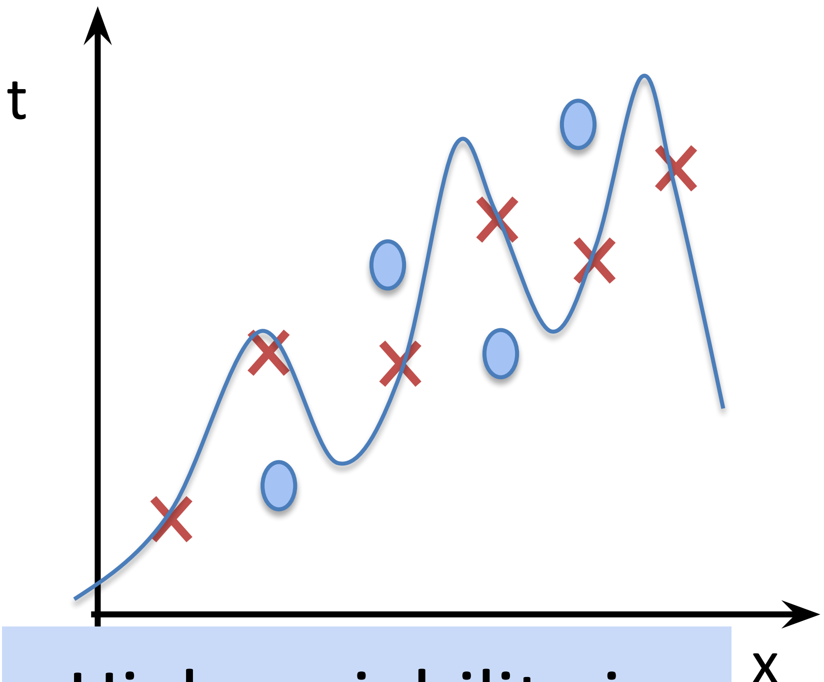


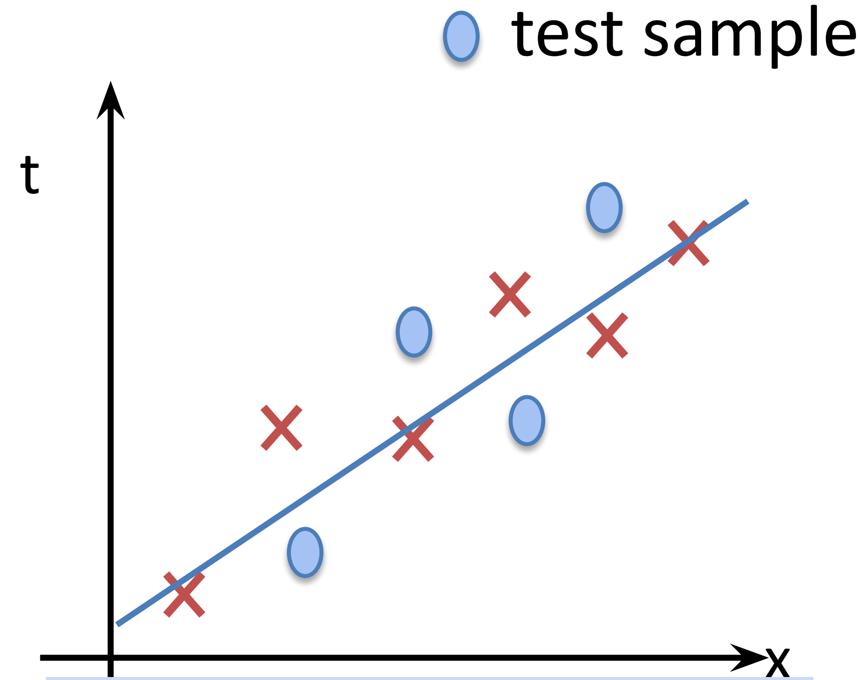
High variability in loss on test data

Low variability in the loss on test data

More reliable on how it performs on unseen test sets

Test data





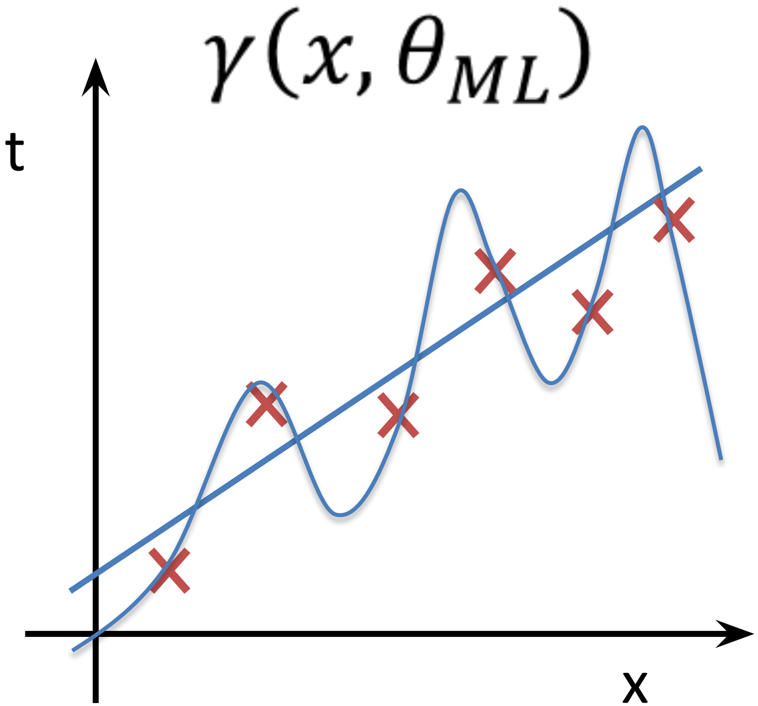
High variability in loss on test data

High variance

Low variability in the loss on test data

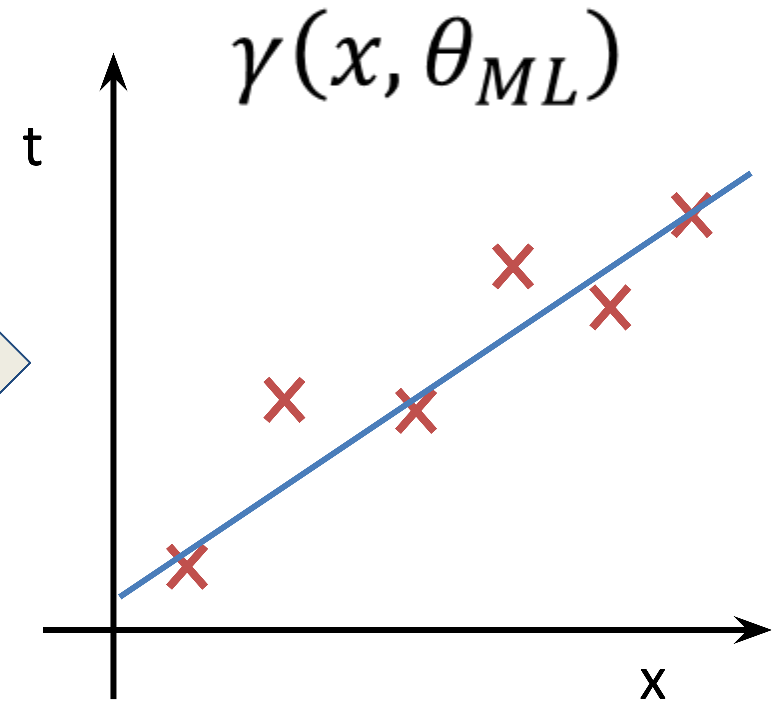
Low variance

What does a regularizer do?



Low training error

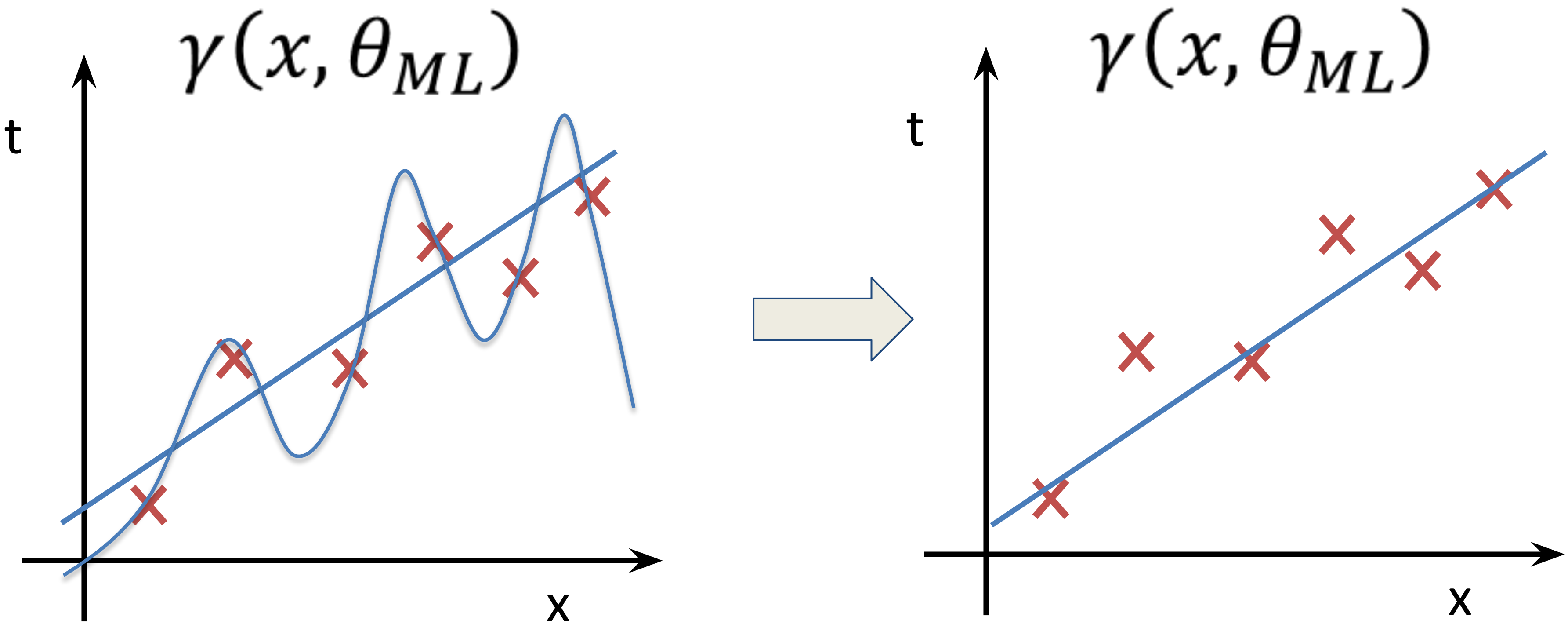
High test error



High training error

Low test error

What does a regularizer do?



Low training error = low bias	High training error = high bias
High test error = high variance	Low test error = low variance

What would be desirable of a good model?

What would be desirable of a good model?

Quiz question · 77 answers · 77 participants

Low bias, high variance - 14 answers — 18%

Low bias, low variance - 57 answers ✓ — 74%

High bias, low variance - 6 answers — 8%

High bias, high variance - 0 answers — 0%

slido

Topics today

- Interesting questions after class
- Loss functions
- **The continued saga of over-fitting**
  - Regularizers
  - Bias and variance
- SVM Intuition and Formulation

Reasons for overfitting

- **What is it:** Model overfits to the underlying training data.

Model too complex	Model not well-trained	Small training data	Train-test data distribution mismatch

Reasons for overfitting

- **What is it:** Model overfits to the underlying training data.

Model too complex	Model not well-trained	Small training data	Train-test data distribution mismatch

⇩

1. Model pruning
2. Regularization

Reasons for overfitting

- **What is it:** Model overfits to the underlying training data.

Model too complex

Model not well-trained

Small training data

Train-test data distribution mismatch

1. Model pruning
2. Regularization

1. Early stopping
2. Ensemble models
3. Better loss functions

Reasons for overfitting

- **What is it:** Model overfits to the underlying training data.

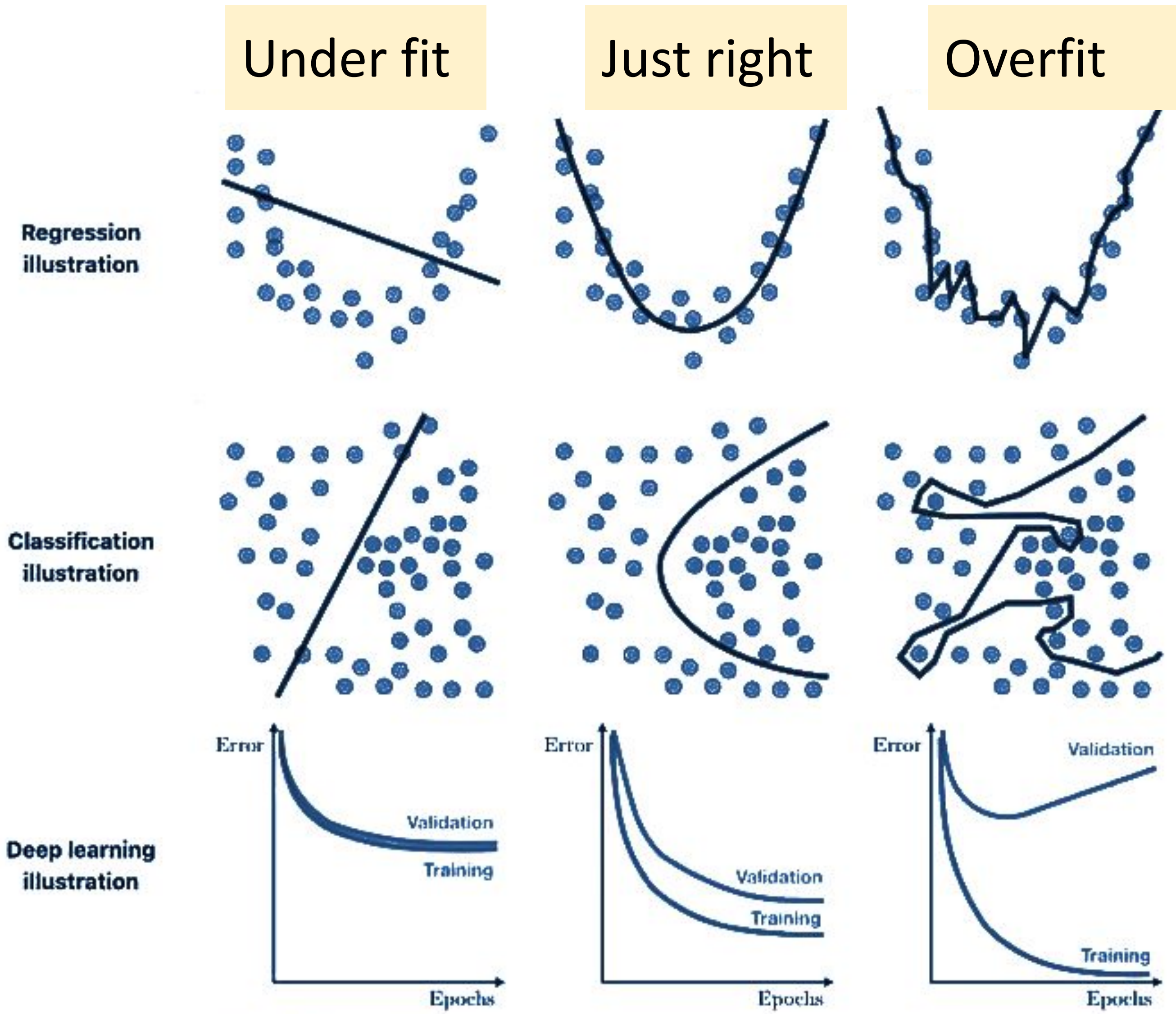
Model too complex	Model not well-trained	Small training data	Train-test data distribution mismatch
1. Model pruning 2. Regularization	1. Early stopping 2. Ensemble models 3. Better loss functions	Data augmentation	

Reasons for overfitting

- **What is it:** Model overfits to the underlying training data.

Model too complex	Model not well-trained	Small training data	Train-test data distribution mismatch
1. Model pruning 2. Regularization	1. Early stopping 2. Ensemble models 3. Better loss functions	Data augmentation	Iterative training

Summary: Underfitting v/s just right v/s over fitting



Topics today

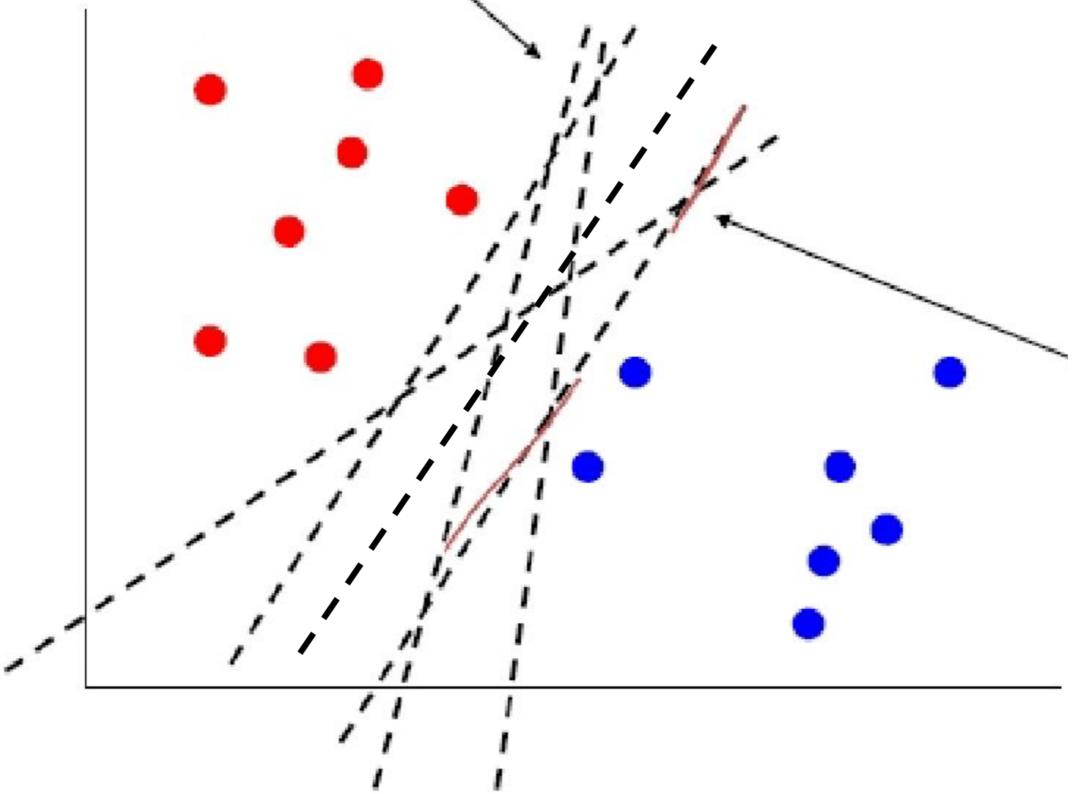
- Interesting questions after class
- Loss functions
- The continued saga of overfitting
  - Regularizers
  - Bias and variance
- **SVM Intuition and Formulation**

Recall: Training Setup

Target Function: $\gamma_i = a^T x_i + b$

Separating two classes

Decision boundary: $a^T x + b = 0$



Many possible lines!

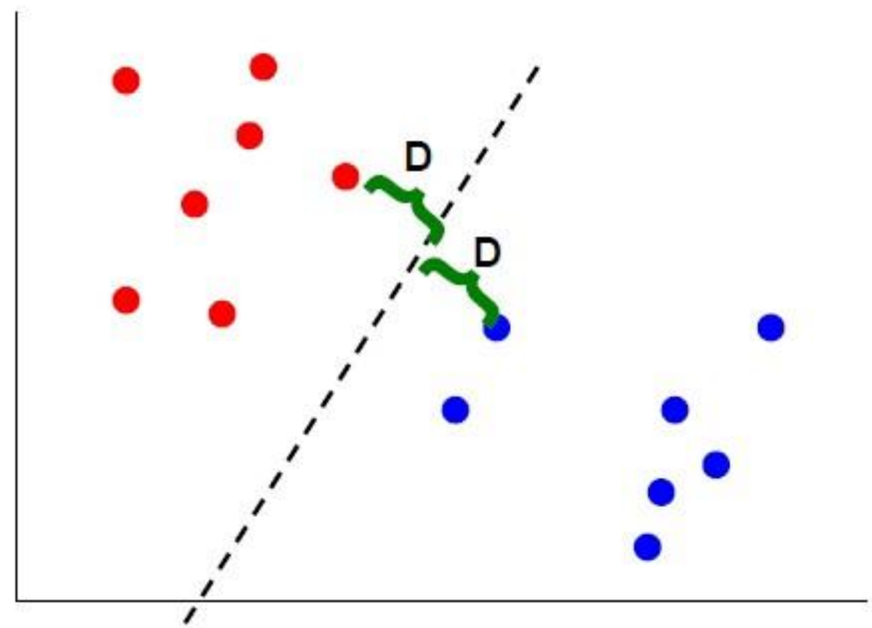
$$y = \begin{cases} +1 \text{ [red] if } \operatorname{sign}(a^T x + b \geq 0) \\ -1 \text{[blue] if } \operatorname{sign}(a^T x + b < 0) \end{cases}$$

Max margin classification

- Instead of fitting all the points, focus on **boundary points**
- **Aim**: learn a boundary that leads to the largest margin (buffer) from points on both sides

- **Why**: robust to small perturbations near the boundary

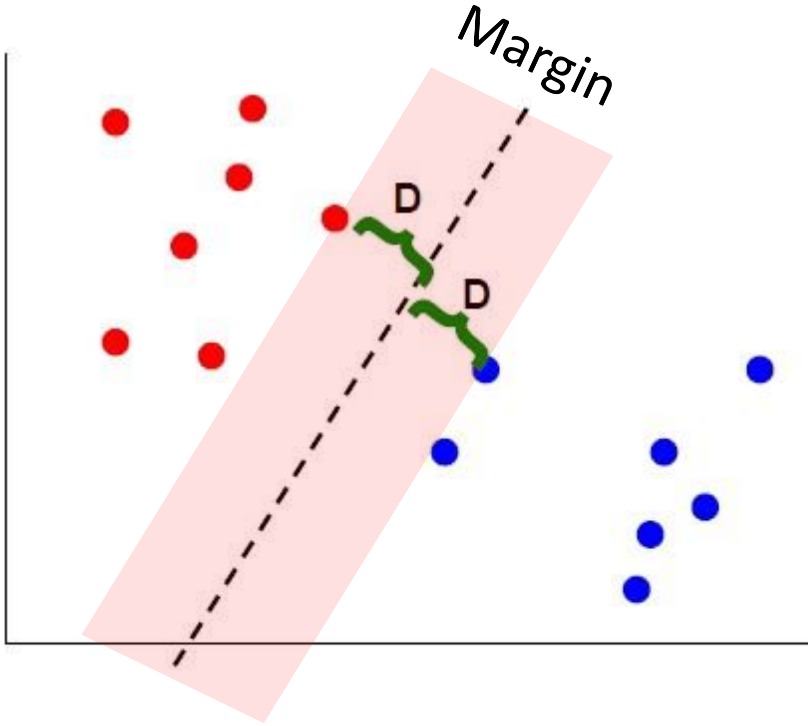
- And works well in practice!



Max margin classification

- Instead of fitting all the points, focus on **boundary points**

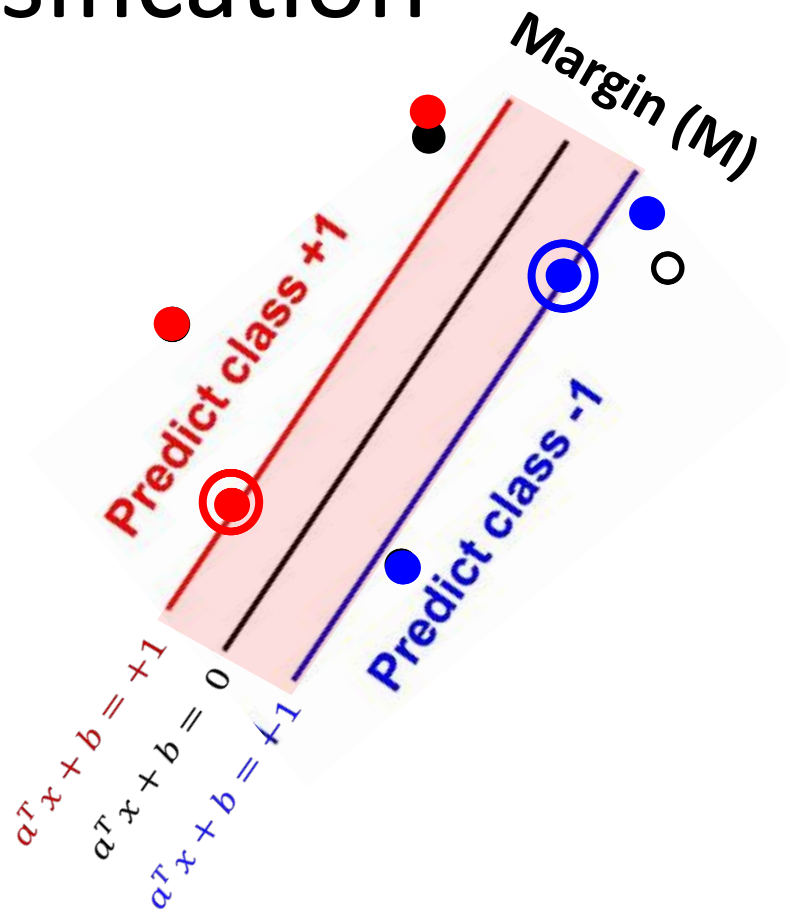
- **Aim**: learn a boundary that leads to the largest margin (buffer) from points on both sides



- **In practice:** Expand the decision boundary to include a **margin** (until we hit first point on either side)?

Max margin classification

- **In practice:** Expand the decision boundary to include a **margin** (until we hit first point on either side)?



Classify as $+1$	if	$a^T x + b \geq +1$
Classify as -1	if	$a^T x + b \leq -1$
Undefined	if	$-1 < a^T x + b < 1$

- Subset of vectors that support (determine boundary) are called the support vectors (circled).

Computing the margin

- For support vectors, $a^T x + b = \pm 1$
- **Distance** from a point and line =

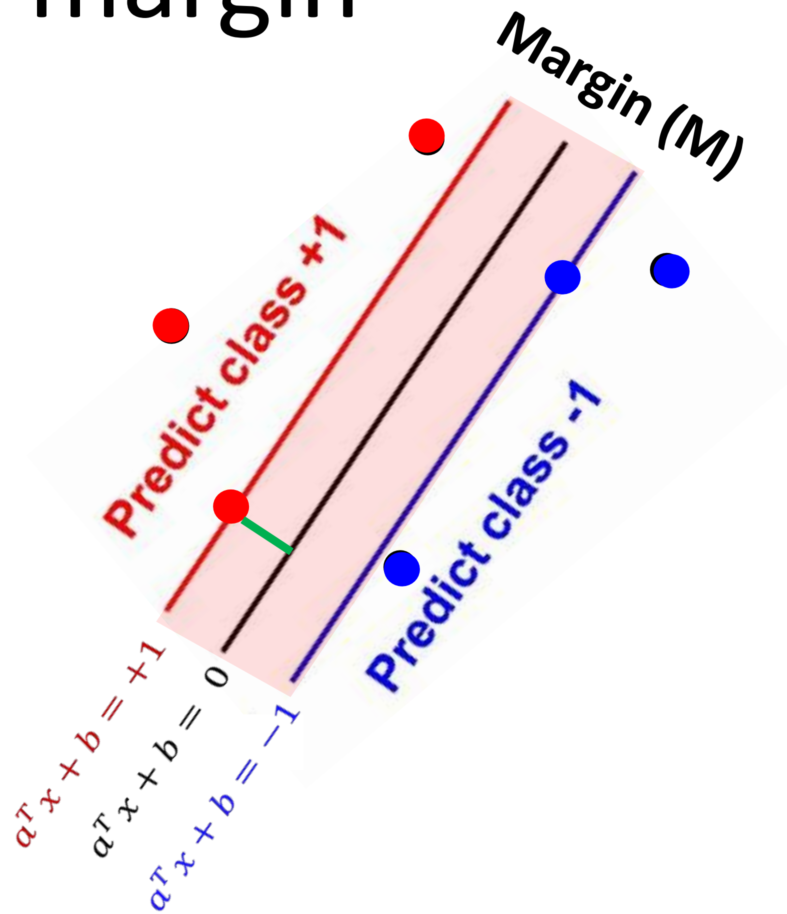
$$\frac{|a^T x + b|}{||a||}$$

- For support vectors,

$$\frac{|a^T x + b|}{||a||} = \frac{\pm 1}{||a||}$$

$$\mathrm{M} = \left|\frac{1}{||a||} - \frac{-1}{||a||}\right|$$

$$\boxed{= \frac{2}{||a||}}$$



Slide inspiration: Kristen Grauman

Finding the maximum margin line

- **Goal:** Maximize margin $\frac{2}{||a||} = \frac{2}{a^T a}$

Classify as $+1$ if $a^T x + b \geq +1$

Classify as -1 if $a^T x + b \leq -1$

Minimize: $\frac{1}{2}\boldsymbol{a}^T\boldsymbol{a}$

Subject to: $\boldsymbol{y}(a^T\boldsymbol{x} + b) \geq 1$

Putting it all together: Linear SVM Formulation

- **Aim**: learn a boundary that leads to the largest margin (buffer) from points on both sides.

Condition1: Maximizes the margin

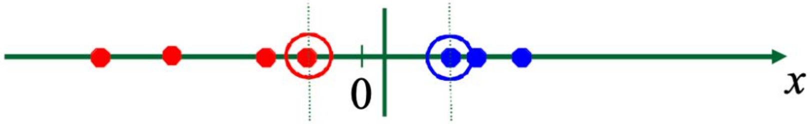
$$\min \ \frac{1}{2}\|a\|^2$$

Condition2: Classifies input data correctly.

$$s.t. \ (a^T x_i + b) y_i \geq 1 \ \ \forall i$$

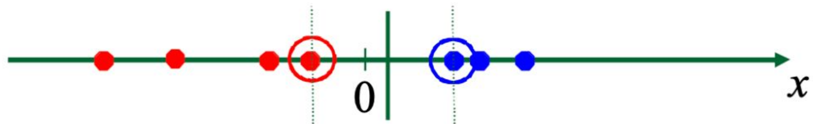
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

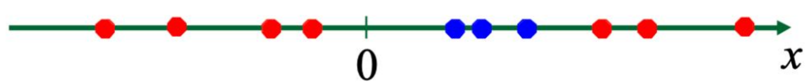


Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

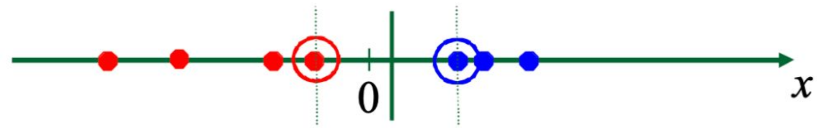


- But what are we going to do if the dataset is just too hard?



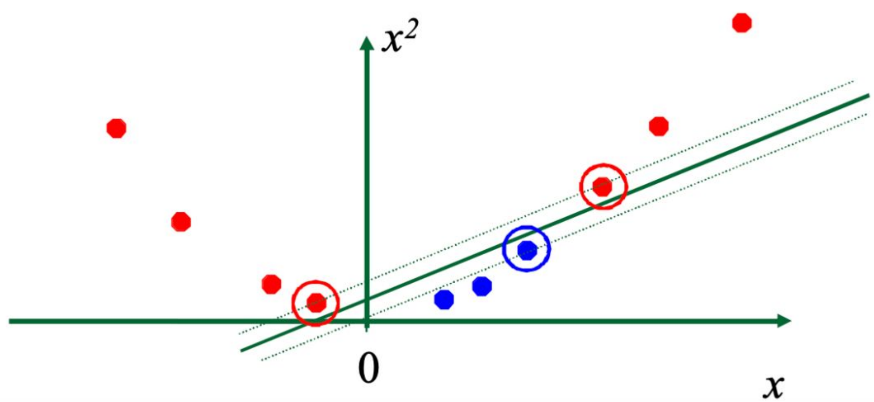
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:



Non-linear SVMs

Step 1: Start with low-dimensional feature space (e.g., 1D)

Step 2: Transform it into higher dimensional feature space (e.g., 2D)

Step 3: Find a max-margin classifier that separates data in this higher dimensional space.

Slide inspiration: Kristen Grauman

Non-linear SVMs

Step 1: Start with low-dimensional feature space (e.g., 1D)

Step 2: Transform it into higher dimensional feature space (e.g., 2D)

Step 3: Find a max-margin classifier that separates data in this higher dimensional space.

Non-linear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

Computed between pairs of points.

Non-linear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i a_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Slide inspiration: Kristen Grauman

Example Kernel functions

- Linear: $$K(x_i, x_j) = x_i^T x_j$$

- Gaussian RBF: $$K(x_i,x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)$$

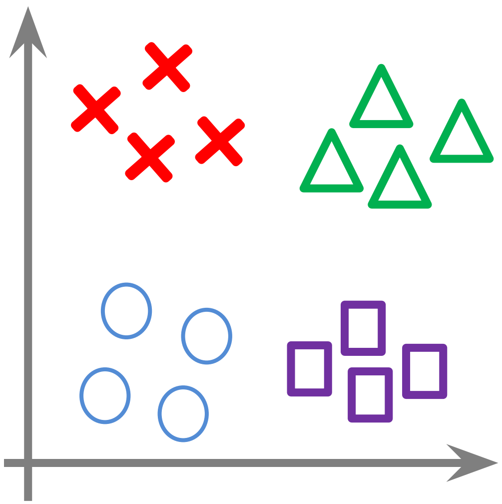
- Histogram intersection:

$$K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k))$$

Theory behind why they work (Mercer's theorem): beyond the scope of this class.

Slide inspiration: Kristen Grauman

Multi-class SVM



$$y \in \{1, 2, 3, \ldots, K\}$$

Many SVM packages already have built-in multi-class classification functionality

Two common methods:

1. **One-vs-all**: train K SVMs that separate one class from the rest, pick the one with the highest score γ
2. **All-vs-all**: train a binary classifier for each pair of classes, pick the class who gets the most votes

Next Class

Random Forests:

Decision tree, information gain, decision forest

Reading: Forsyth Ch 2.2

Gradient Descent Algorithm

Notation

$$u = [a, b]$$

$$\nabla g = \begin{pmatrix} \frac{\partial g}{\partial u_1} \\ \frac{\partial g}{\partial u_2} \\ \cdots \\ \frac{\partial g}{\partial u_d} \end{pmatrix}$$

Set $u = 0$

Repeat {

$$u_j := u_j - \eta \nabla g(u)$$

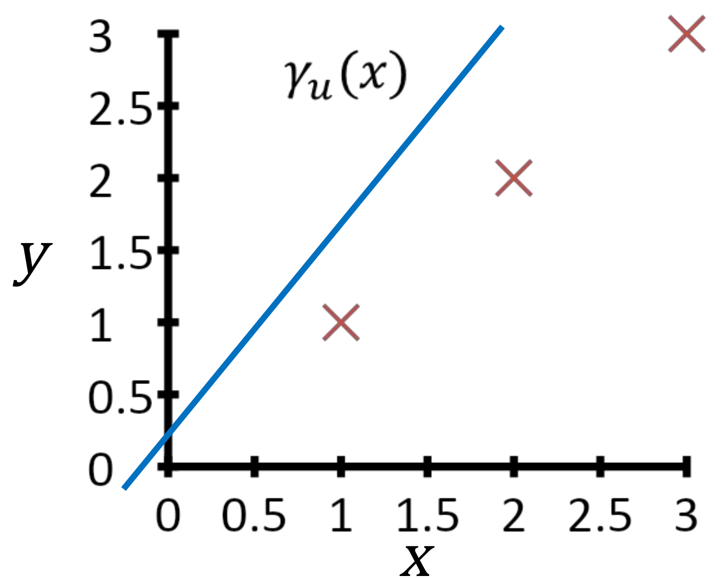
simultaneously for all $j = 0, \dots, d$

} until convergence

Gradient Descent: Intuition

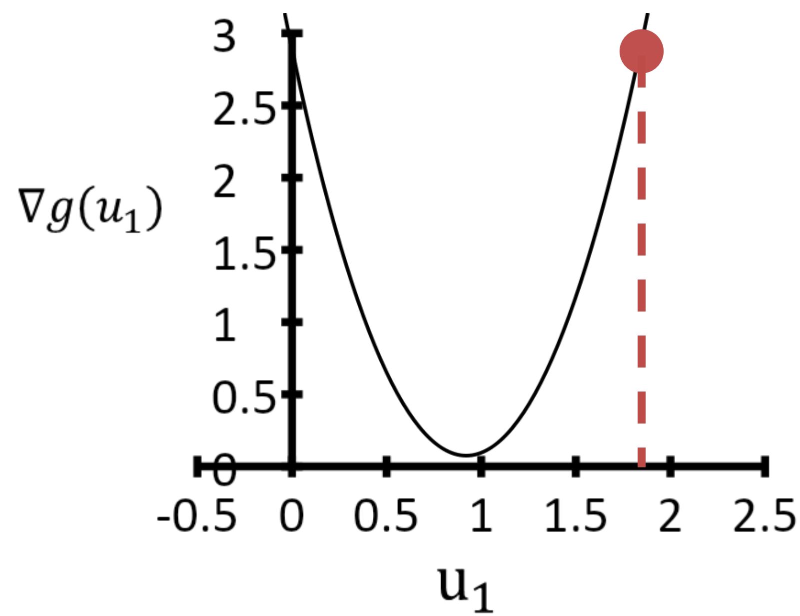
$\gamma_u(x)$

(for fixed u_1, this is a function of x)



$\nabla g(u_1)$

(function of the parameter u_1)



Gradient Descent Update: $u_j := u_j - \eta \nabla g$

slido

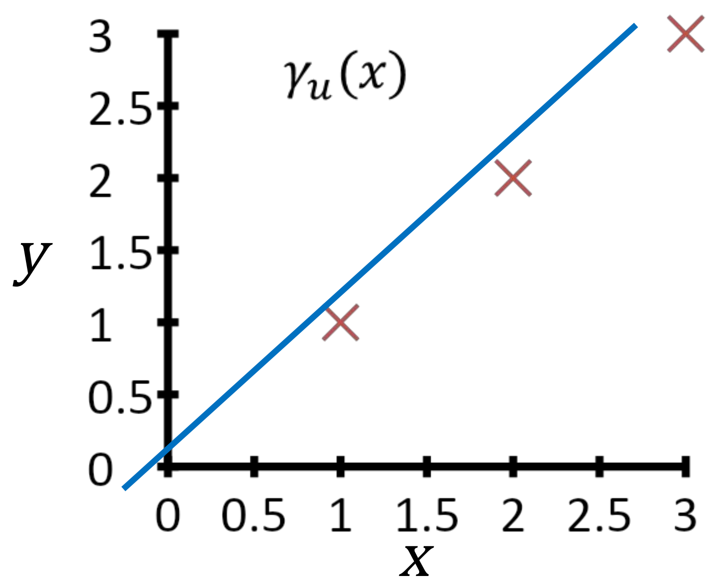
What happens if your learning rate is very high?

ⓘ Start presenting to display the poll results on this slide.

Gradient Descent: Intuition

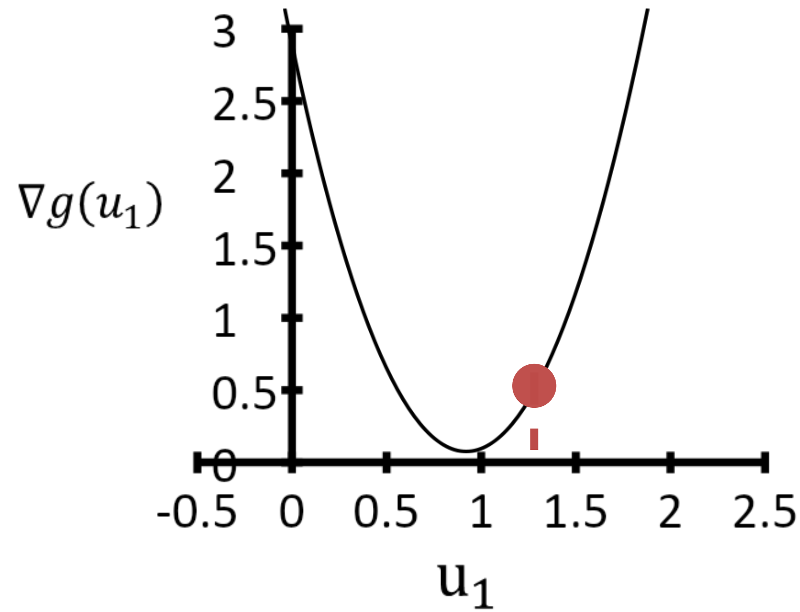
$\gamma_u(x)$

(for fixed u_1, this is a function of x)

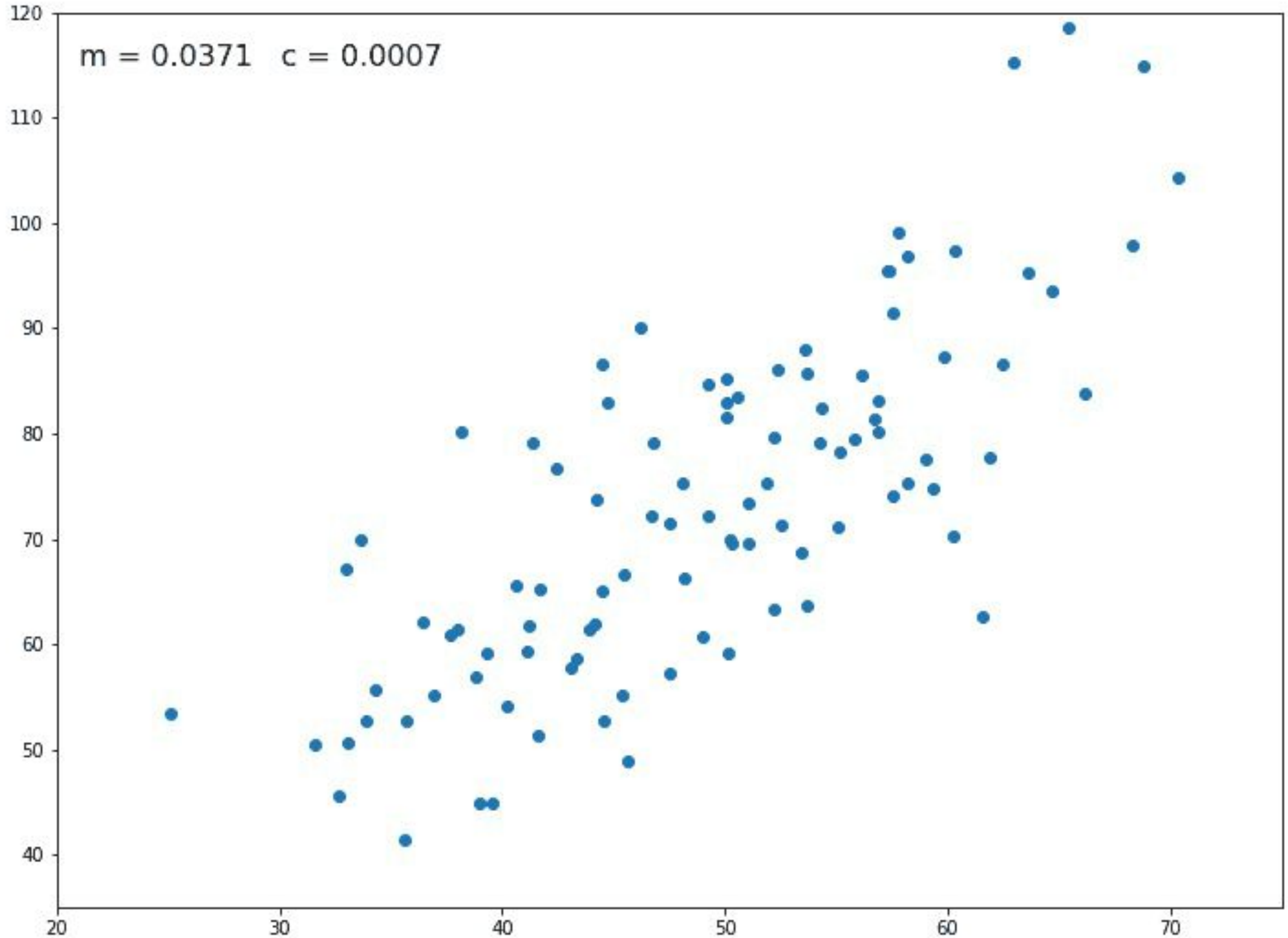


$\nabla g(u_1)$

(function of the parameter u_1)



Gradient Descent Update: $u_j := u_j - \eta \nabla g$



Gradient descent illustration (credit: https://towardsdatascience.com/)

What if our dataset was <u>very</u> large?

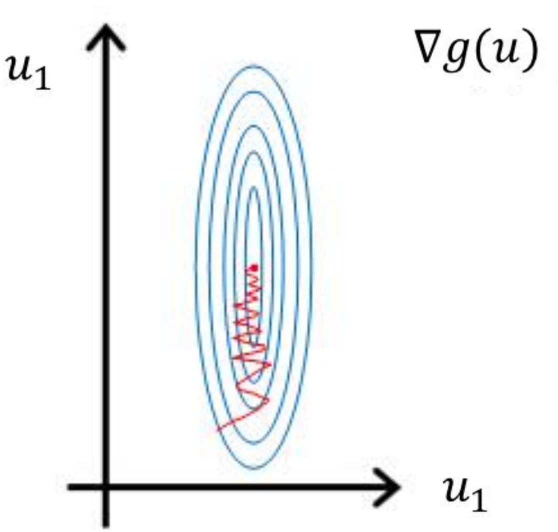
- Use stochastic gradient descent!
- **Step 1:** Select a (random) batch of N_b data samples
- **Step 2:** Use gradient descent algorithm on the batch to take a step

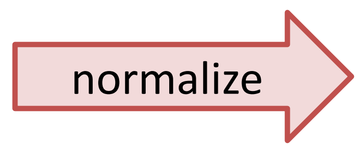
$$\mathrm{p}_{N_b}^{n} = -\sum_{j \in \mathrm{batch}} \nabla g_i(u)$$

$$u^{(n+1)} = u^n + \eta p_{N_b}^{n}$$

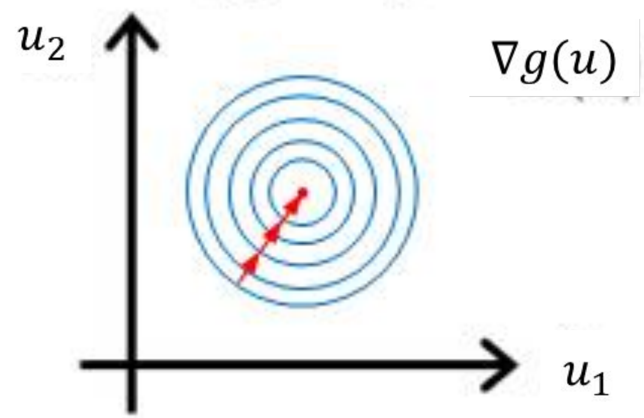
Feature normalization

- If features have very different scale, GD can get "stuck" since x_j affects size of gradient in the direction of j[th] dimension
- Normalizing features to be zero-mean (μ) and same-variance (σ) helps gradient descent converge faster





$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$



Which method would be faster to train?

ⓘ Start presenting to display the poll results on this slide.

slido

Which method would be faster at inference time?

ⓘ Start presenting to display the poll results on this slide.

Bayesian Modeling: Notation in the textbook

- $\mathbf{x}_{\mathbf{tr}}$: observed data
- $\mathbf{y}$: target labels
- Given a new test sample $x_{te,}$
- $p(y|x_{te})$: probability of belonging to a particular class.
- Goal: Learn $p(y|\mathbf{x}_{\mathbf{tr}})$ on observed data.
- How? Use Bayes' rule

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

Bayesian Modeling

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

Your estimate from the observed data

Your mental model of the target distribution

Bayesian Modeling

Assumption: every datapoint is an IID (Independent and identically distributed random variables).

$$p(\mathbf{x}|y) = \prod_j p(x^{(j)}|y).$$

$$
\begin{aligned}
p(y|\mathbf{x}) &= \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \\
&= \frac{\left(\prod_j p(x^{(j)}|y)\right)p(y)}{p(\mathbf{x})} \\
&\propto \left(\prod_j p(x^{(j)}|y)\right)p(y).
\end{aligned}
$$

Apply maximum likelihood estimate (MLE). Pick

Are we assuming a prior over model parameters or target labels?

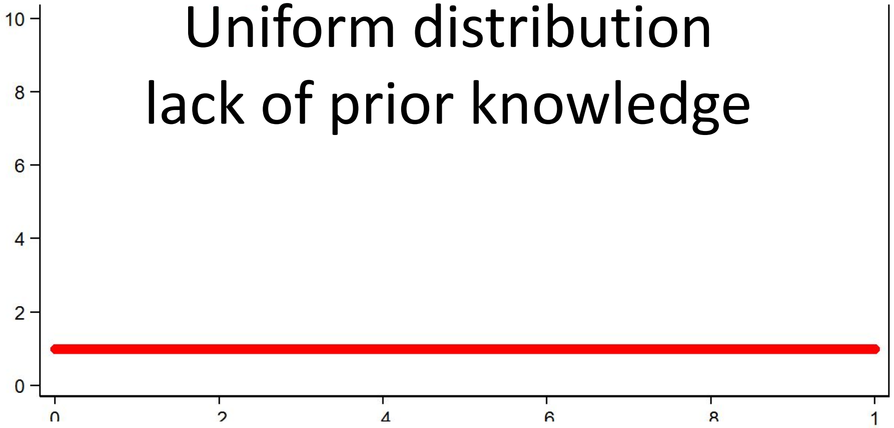
From the textbook $$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

MLE = $\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(u^{(i)}|\theta)$

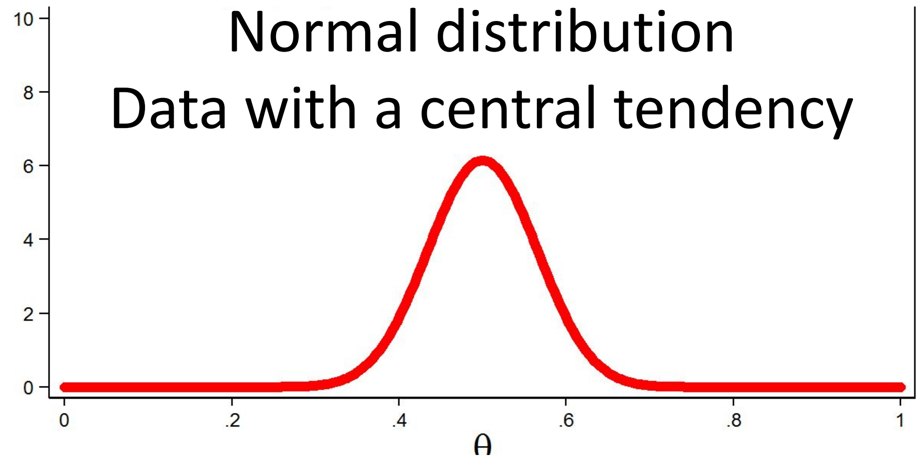
- We are assuming a prior over the parameters of the distribution we wish to fit to the outcomes.
- The notation could be confusing.

Can we assume a prior over model parameters?

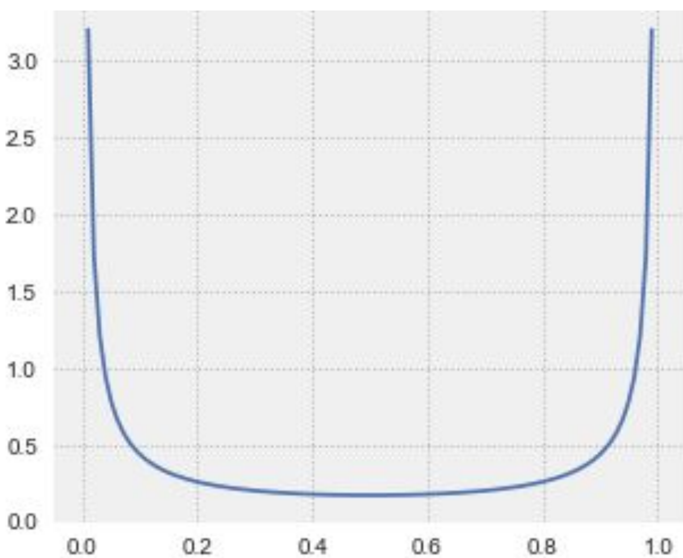
Uniform distribution
lack of prior knowledge



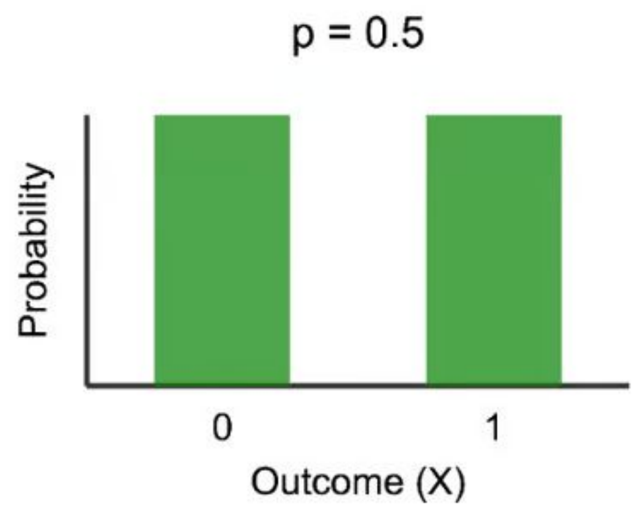
Normal distribution
Data with a central tendency



We know that outcome falls between 0 and 1



Bernoulli distribution
(two outcomes)



PSET-1 Clarification

```
def question_2(arr: np.ndarray) -> np.ndarray:
"""
Given a list of padded entries with pad class 100.
"""
# Write your code in this block -----------------------------------------------------------


#First identify the shape

rows,columns=arr.shape

array=np.zeros((rows,columns,101))
array[:,list(range(columns)),arr[:]]=1


# End of your code ------------------------------------------------------------------------
return array
```

- You can assume that the sentences are padded before it's passed onto the question_2 function.

- Goal: create a one-hot representation of each word in the sentence accounting for the pad token. The expected output will be something like (Sentence,num_words,vocabulary).

Any follow ups? Look at previous posts. If not answered, then post them on piazza