Linear classifiers:
Overfitting & regularization

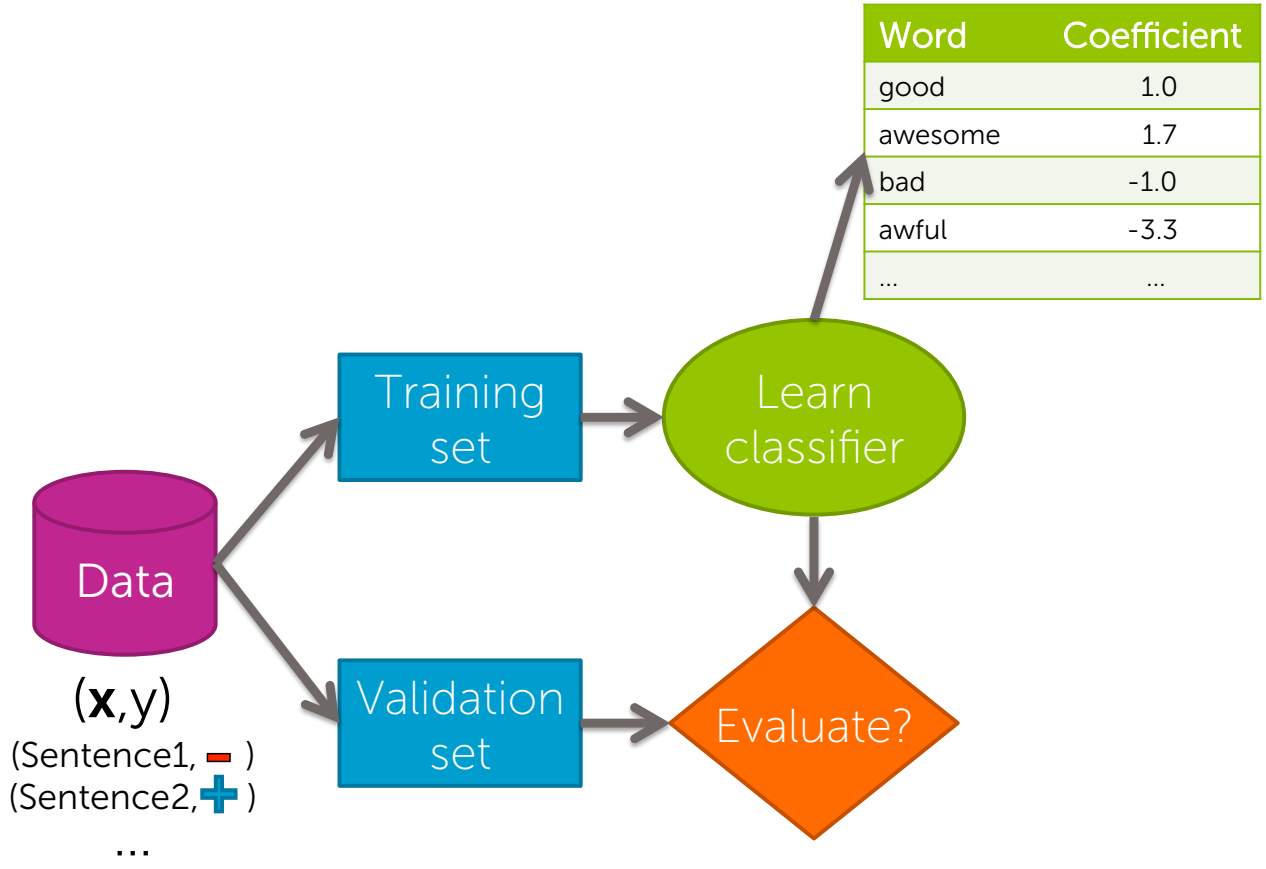
Emily Fox & Carlos Guestrin

Machine Learning Specialization

University of Washington

Training and evaluating a classifier

Training a classifier = Learning the coefficients



Classification error

Learned classifier

$\hat{y}$ =

Test example

(

)

Mistake!

Correct	0
Mistakes	1

Hide label

Classification error & accuracy

- Error measures fraction of mistakes

$$\text{error} = \frac{\#\ \text{Mistakes}}{\text{Total number of data points}}$$

  - Best possible value is 0.0

- Often, measure **accuracy**
  - Fraction of correct predictions

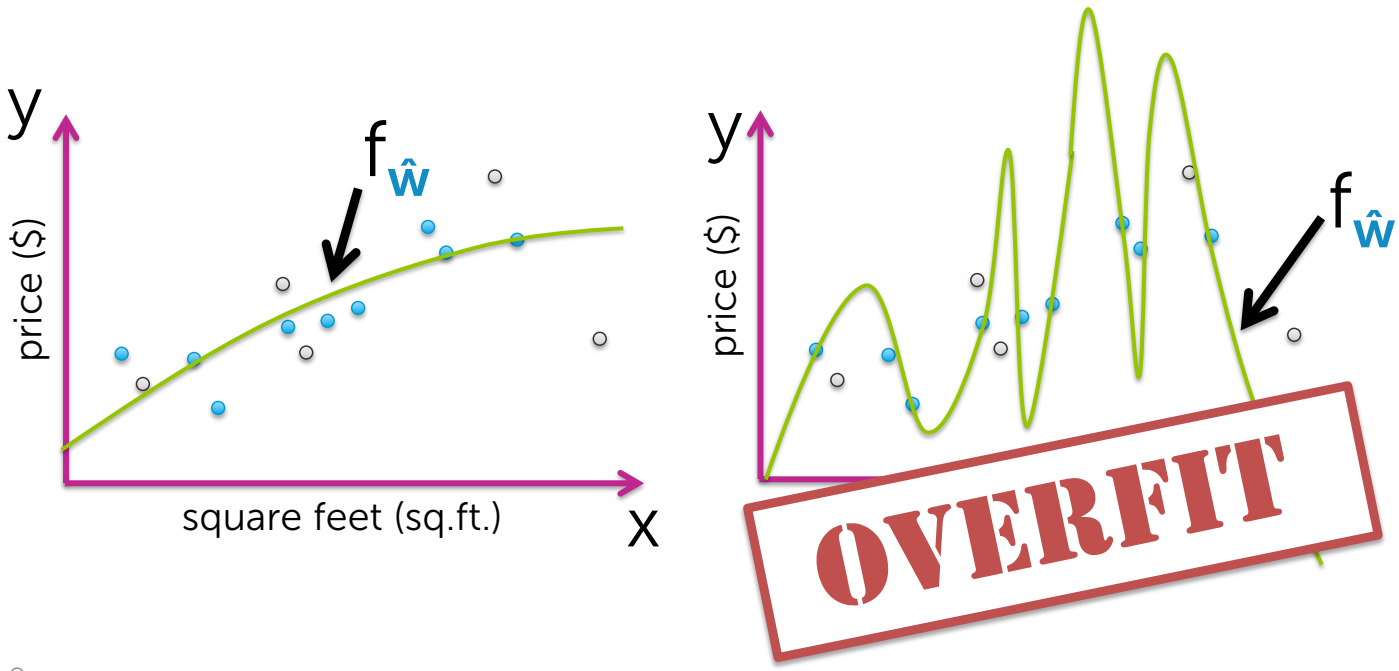
$$\text{accuracy} = \frac{\#\ \text{Correct}}{\text{Total number of data points}}$$

  - Best possible value is 1.0

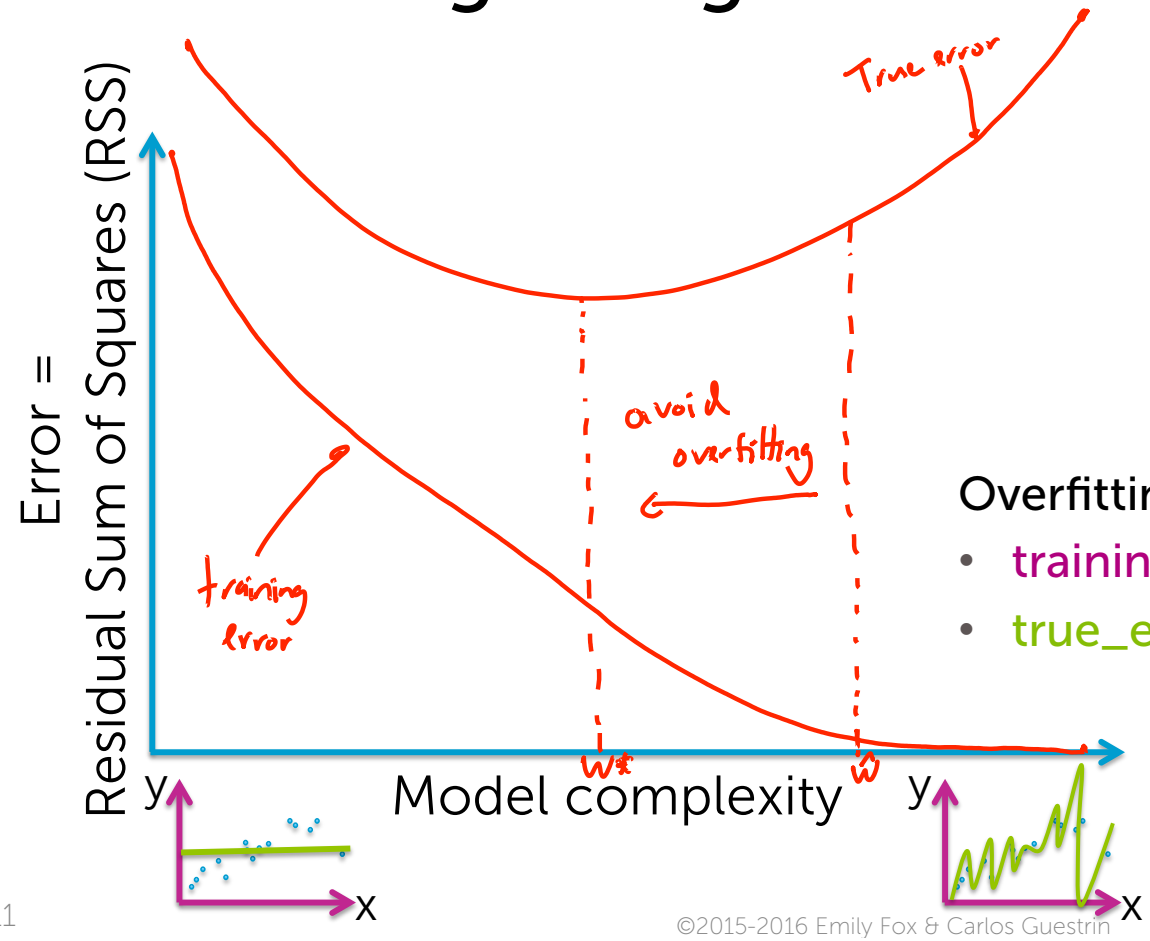
Overfitting in regression: review

Flexibility of high-order polynomials

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \varepsilon_i$$



Overfitting in regression



Overfitting if there exists w^*:

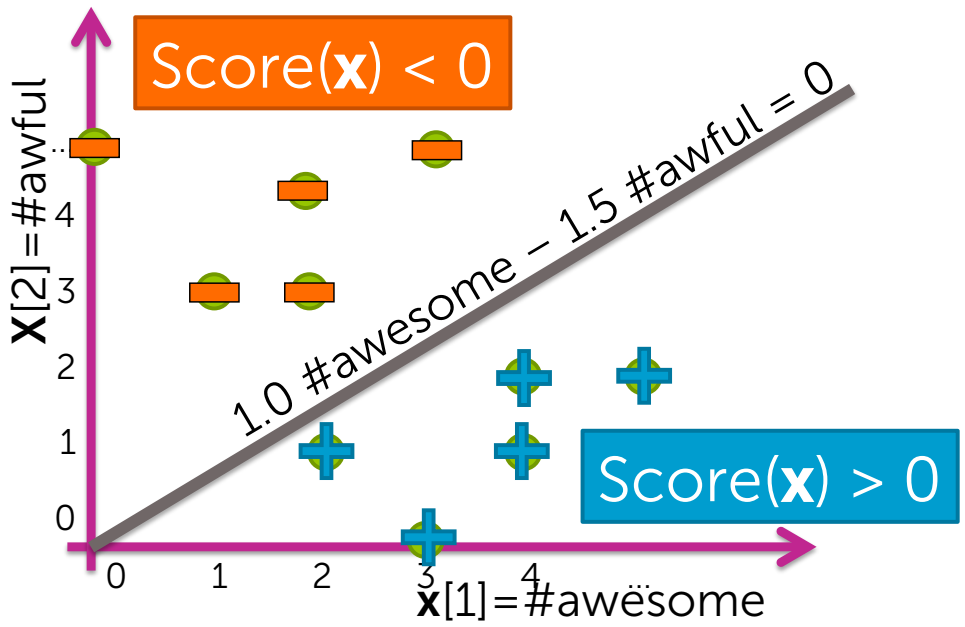
- training_error(w^*) > training_error($\hat{w}$)
- true_error(w^*) < true_error($\hat{w}$)

Overfitting in classification

Decision boundary example

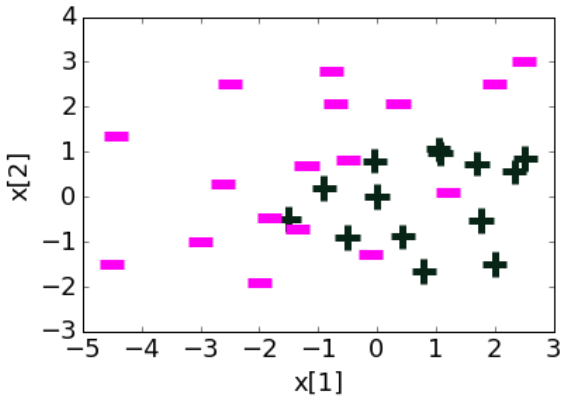
Word	Coefficient
#awesome	1.0
#awful	-1.5

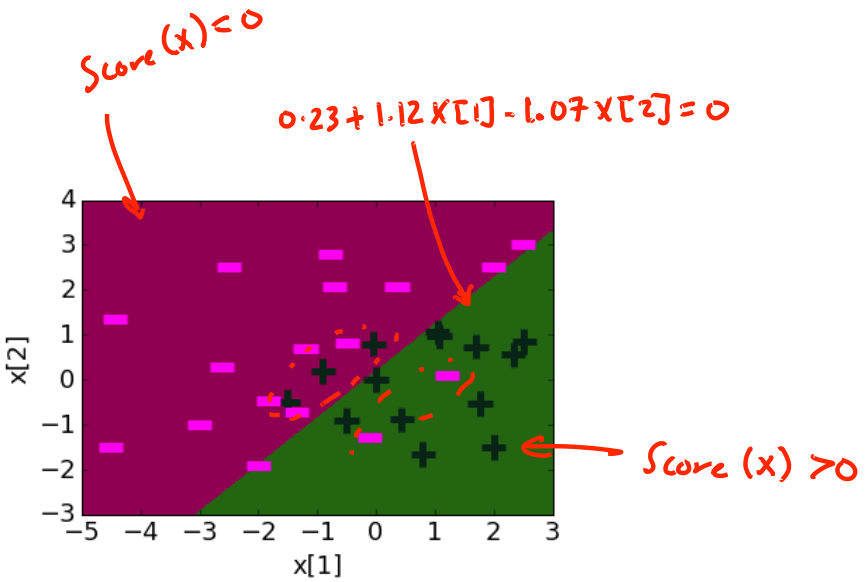
Score(x) = 1.0 #awesome – 1.5 #awful



Learned decision boundary

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	w_0 1	0.23
$h_1(\mathbf{x})$	w_1 x[1]	1.12
$h_2(\mathbf{x})$	w_2 x[2]	-1.07

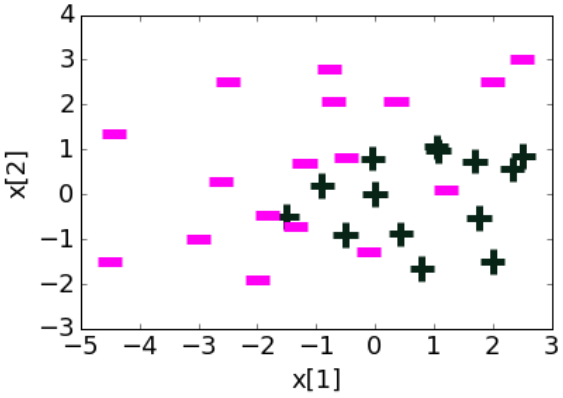


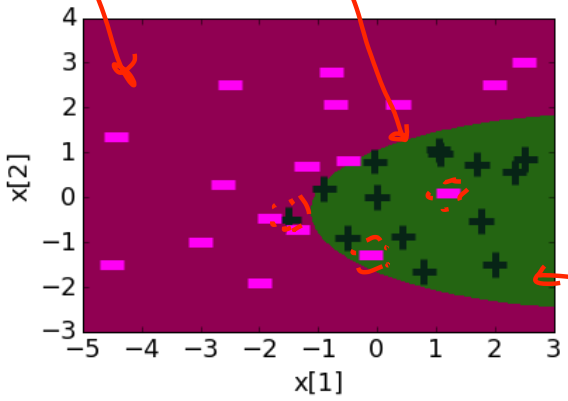


Quadratic features (in 2d)

Note: we are not including cross terms for simplicity

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	1.68
$h_1(\mathbf{x})$	$\mathbf{x}[1]$	1.39
$h_2(\mathbf{x})$	$\mathbf{x}[2]$	-0.59
$h_3(\mathbf{x})$	$(\mathbf{x}[1])^2$	-0.17
$h_4(\mathbf{x})$	$(\mathbf{x}[2])^2$	-0.96





Score(x) < 0

$1.68 + 1.39\,x[1] - 0.59\,x[2] - 0.17(x[1])^2 - 0.96(x[2])^2 = 0$

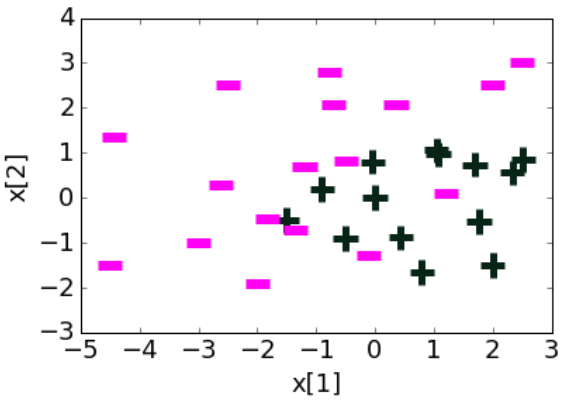
Score(x) > 0

Degree 6 features (in 2d)

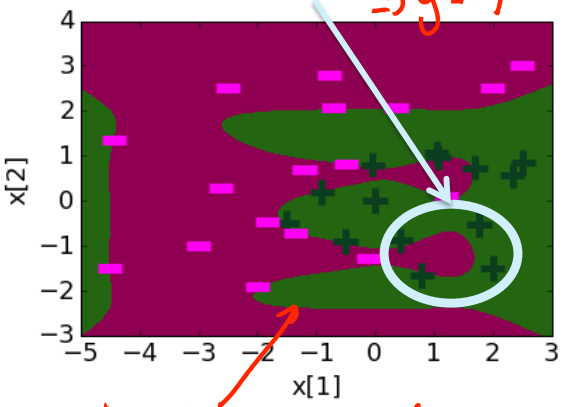
Note: we are not including cross terms for simplicity

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	21.6
$h_1(\mathbf{x})$	$\mathbf{x}[1]$	5.3
$h_2(\mathbf{x})$	$\mathbf{x}[2]$	-42.7
$h_3(\mathbf{x})$	$(\mathbf{x}[1])^2$	-15.9
$h_4(\mathbf{x})$	$(\mathbf{x}[2])^2$	-48.6
$h_5(\mathbf{x})$	$(\mathbf{x}[1])^3$	-11.0
$h_6(\mathbf{x})$	$(\mathbf{x}[2])^3$	67.0
$h_7(\mathbf{x})$	$(\mathbf{x}[1])^4$	1.5
$h_8(\mathbf{x})$	$(\mathbf{x}[2])^4$	48.0
$h_9(\mathbf{x})$	$(\mathbf{x}[1])^5$	4.4
$h_{10}(\mathbf{x})$	$(\mathbf{x}[2])^5$	-14.2
$h_{11}(\mathbf{x})$	$(\mathbf{x}[1])^6$	0.8
$h_{12}(\mathbf{x})$	$(\mathbf{x}[2])^6$	-8.6

Coefficient values getting large



Score($\mathbf{x}$) < 0



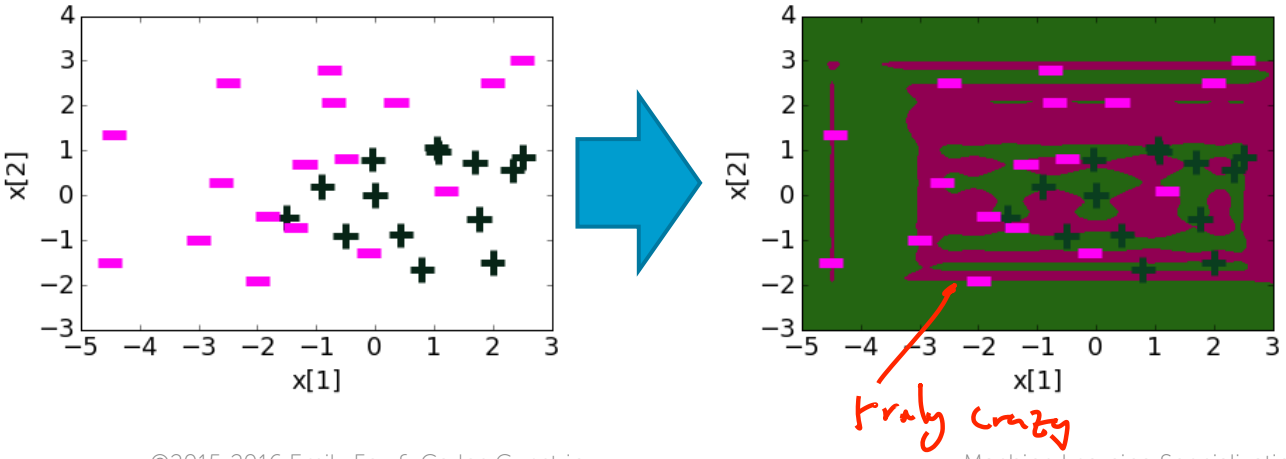
extremely complex (crazy) decision boundary

Degree 20 features (in 2d)

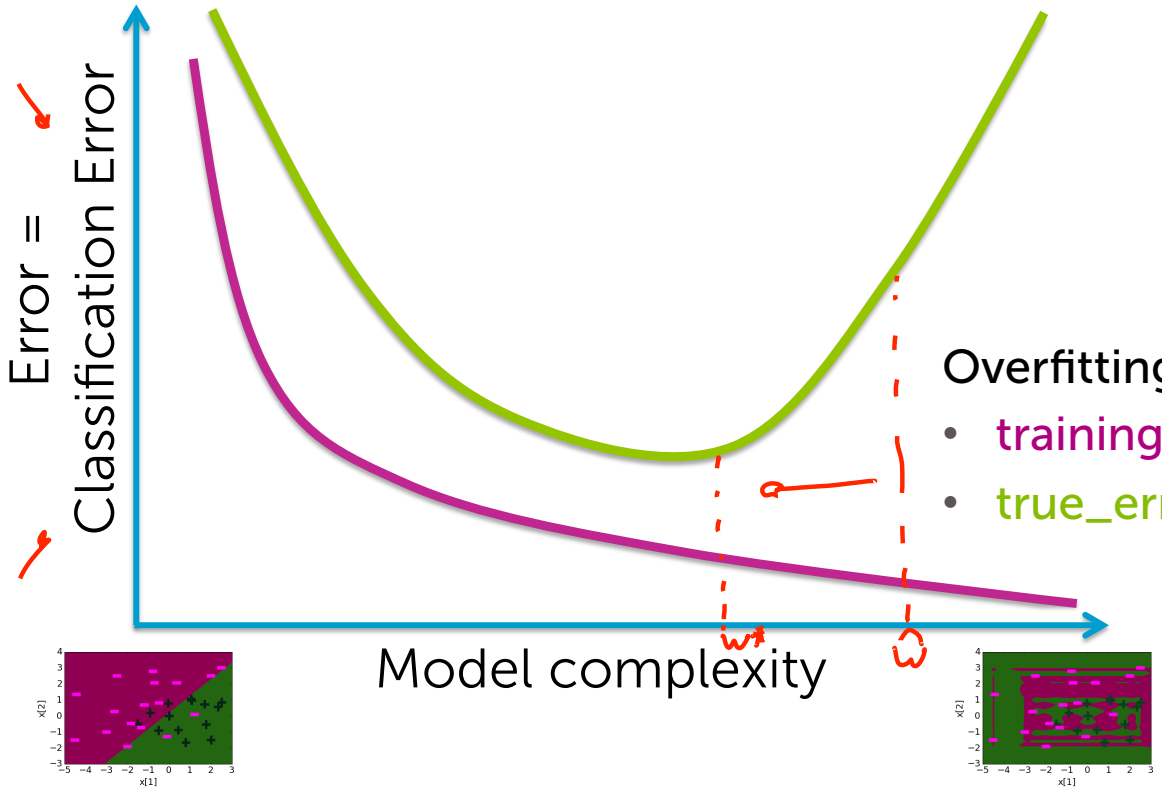
Note: we are not including cross terms for simplicity

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	8.7
$h_1(\mathbf{x})$	$\mathbf{x}[1]$	5.1
$h_2(\mathbf{x})$	$\mathbf{x}[2]$	78.7
...	...	...
$h_{11}(\mathbf{x})$	$(\mathbf{x}[1])^6$	-7.5
$h_{12}(\mathbf{x})$	$(\mathbf{x}[2])^6$	**3803**
$h_{13}(\mathbf{x})$	$(\mathbf{x}[1])^7$	~~21.1~~
$h_{14}(\mathbf{x})$	$(\mathbf{x}[2])^7$	**-2406**
...	...	...
$h_{37}(\mathbf{x})$	$(\mathbf{x}[1])^{19}$	$-2*10^{-6}$
$h_{38}(\mathbf{x})$	$(\mathbf{x}[2])^{19}$	-0.15
$h_{39}(\mathbf{x})$	$(\mathbf{x}[1])^{20}$	$-2*10^{-8}$
$h_{40}(\mathbf{x})$	$(\mathbf{x}[2])^{20}$	0.03

Often, overfitting associated with very large estimated coefficients $\hat{\mathbf{w}}$



Overfitting in classification

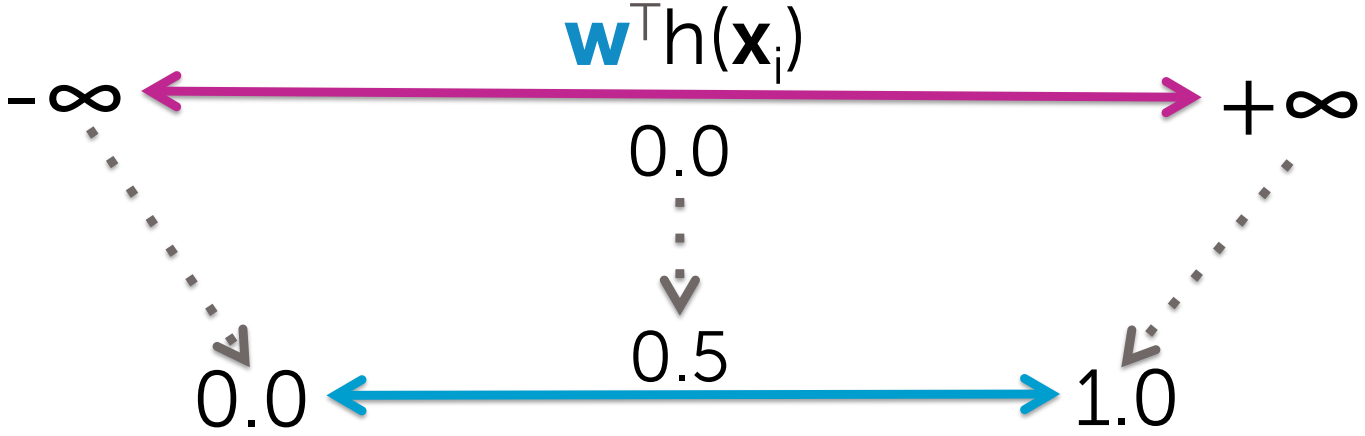


Overfitting if there exists w^*:

- training_error(w^*) > training_error($\hat{w}$)
- true_error(w^*) < true_error($\hat{w}$)

Overfitting in classifiers → Overconfident predictions

Logistic regression model



$$P(y=+1|\mathbf{x}_i,\mathbf{w}) = \text{sigmoid}(\mathbf{w}^T h(\mathbf{x}_i))$$

The subtle (negative) consequence of overfitting in logistic regression

Overfitting ➔ Large coefficient values

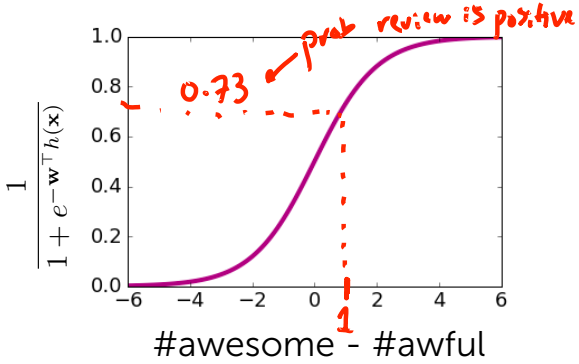
$\hat{\mathbf{w}}^T h(\mathbf{x}_i)$ is very positive (or very negative)
➔ **sigmoid**($\hat{\mathbf{w}}^T h(\mathbf{x}_i)$) goes to 1 (or to 0)

Model becomes extremely overconfident of predictions

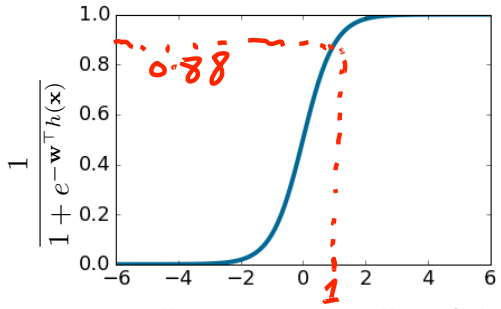
Effect of coefficients on logistic regression model

Input **x**: #awesome=2, #awful=1

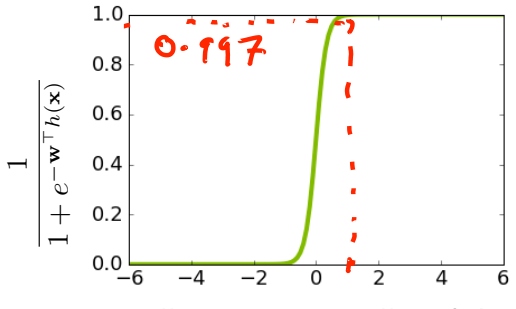
w_0	0
$w_{\#awesome}$	+1
$w_{\#awful}$	-1



w_0	0
$w_{\#awesome}$	+2
$w_{\#awful}$	-2



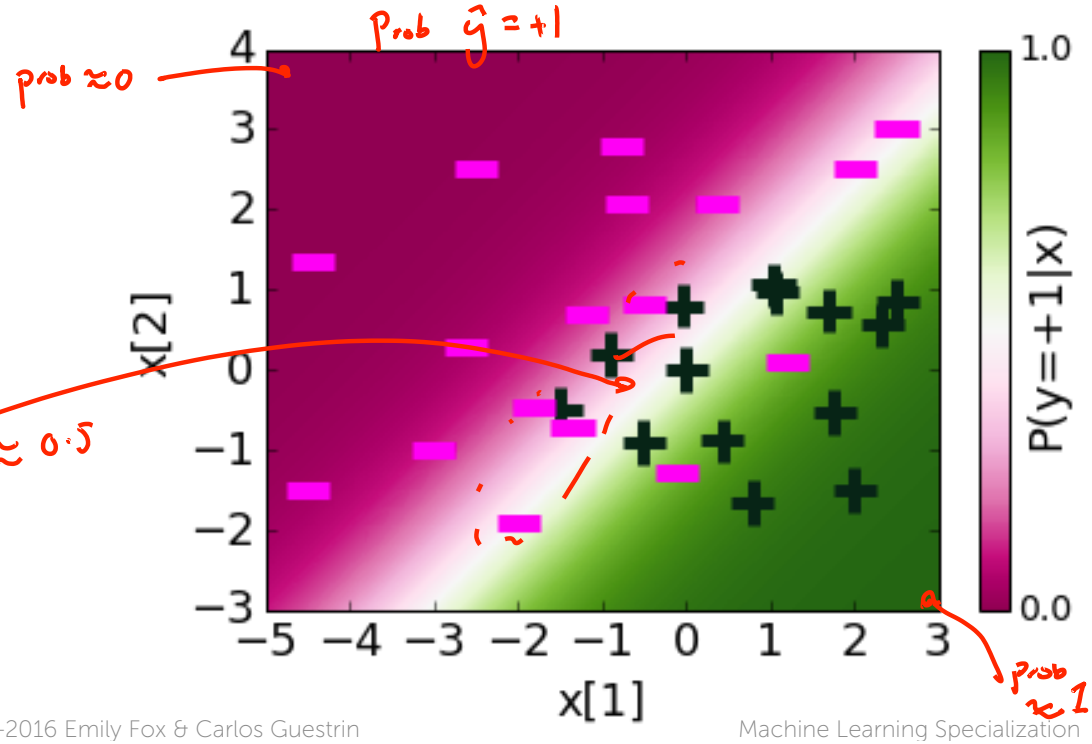
w_0	0
$w_{\#awesome}$	+6
$w_{\#awful}$	-6



Learned probabilities

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	0.23
$h_1(\mathbf{x})$	$\mathbf{x}[1]$	1.12
$h_2(\mathbf{x})$	$\mathbf{x}[2]$	-1.07

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}}$$



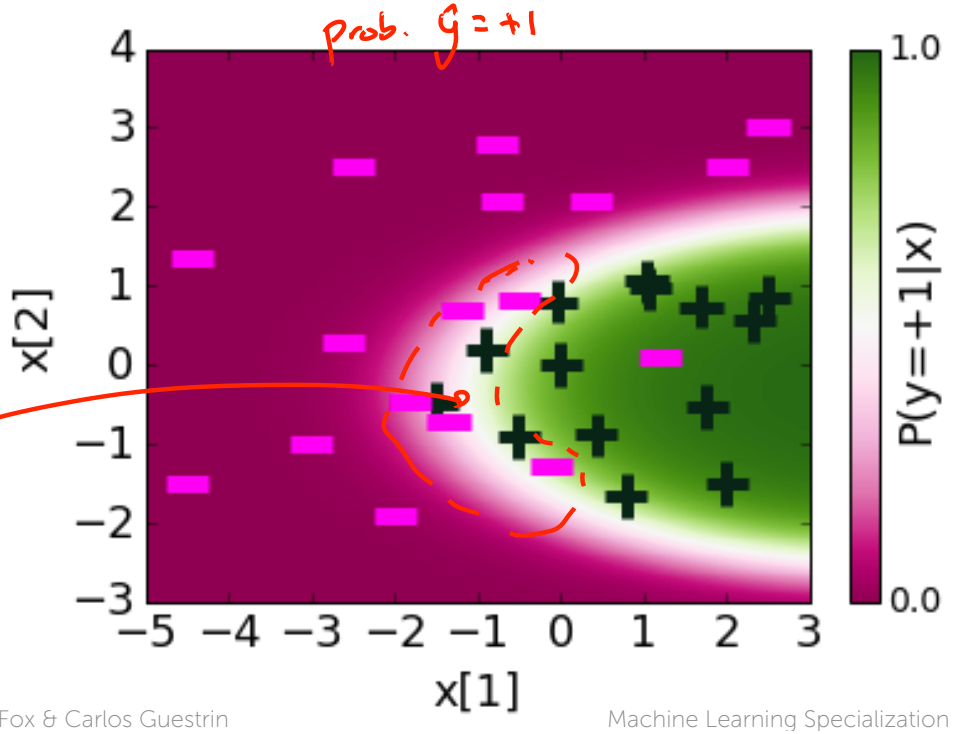
Quadratic features: Learned probabilities

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	1.68
$h_1(\mathbf{x})$	$\mathbf{x}[1]$	1.39
$h_2(\mathbf{x})$	$\mathbf{x}[2]$	-0.58
$h_3(\mathbf{x})$	$(\mathbf{x}[1])^2$	-0.17
$h_4(\mathbf{x})$	$(\mathbf{x}[2])^2$	-0.96

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}}$$

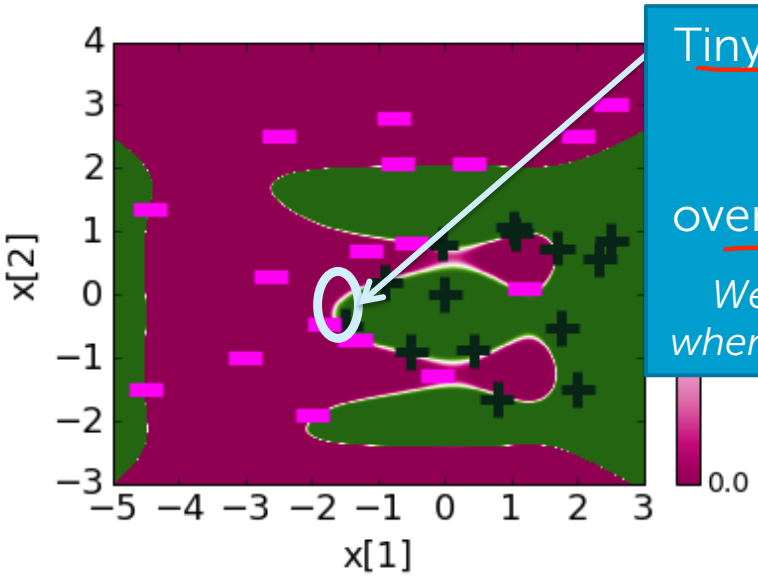
better fit to data

uncertainty region narrower



Overfitting ➔ Overconfident predictions

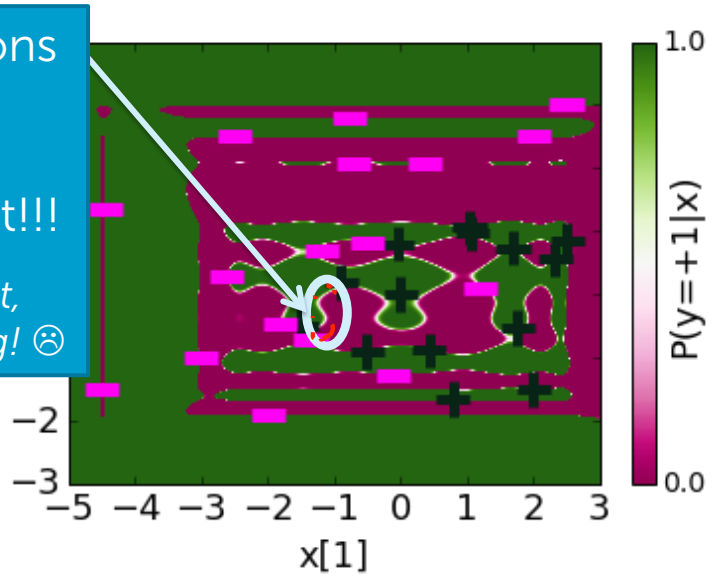
Degree 6: Learned probabilities



Tiny uncertainty regions
➔
Overfitting & overconfident about it!!!

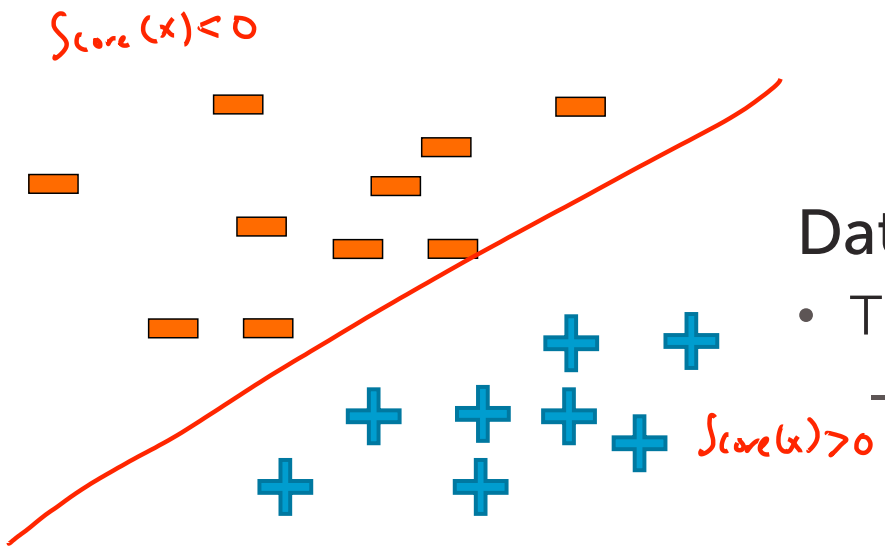
We are sure we are right, when we are surely wrong! ☹

Degree 20: Learned probabilities



Overfitting in logistic regression: Another perspective

Linearly-separable data



Data are linearly separable if:

- There exist coefficients $\hat{w}$ such that:
  - For all positive training data

$$Score(x) = \hat{w}^T h(x) > 0$$

  - For all negative training data

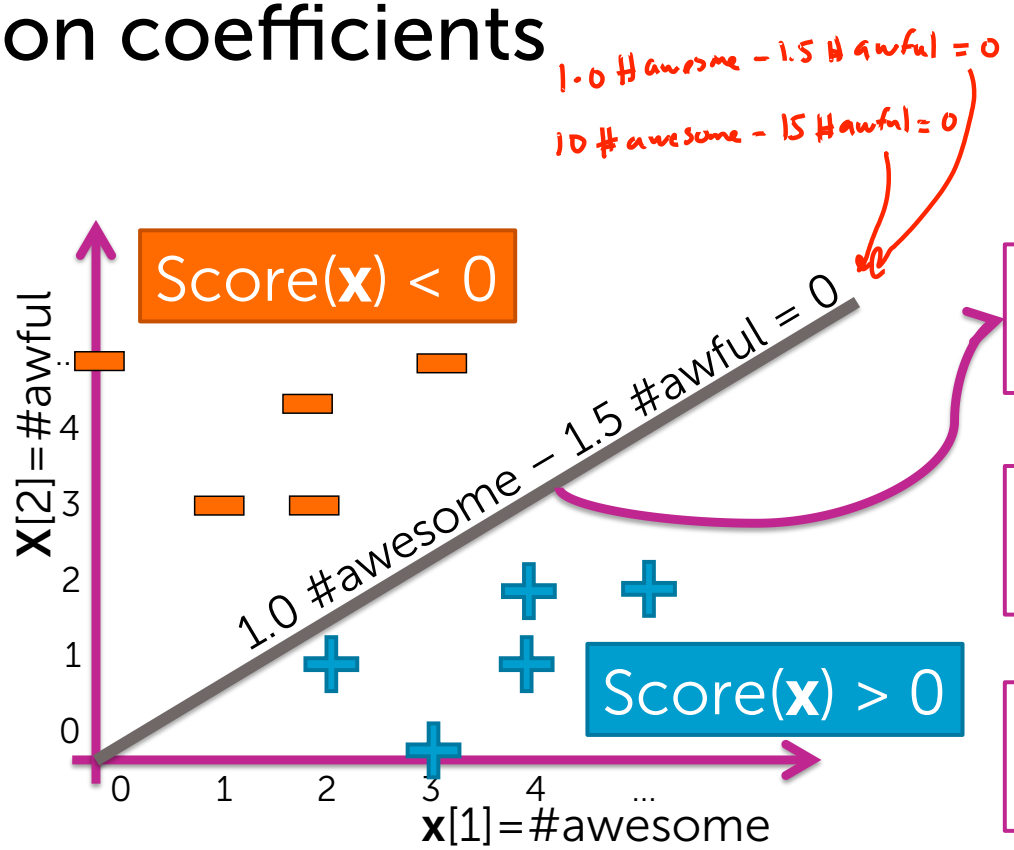
$$Score(x) = \hat{w}^T h(x) < 0$$

Note 1: If you are using D features, linear separability happens in a D-dimensional space

Note 2: If you have enough features, data are (almost) always linearly separable

$$\text{training\_error}(\hat{w}) = 0$$

Effect of linear separability on coefficients

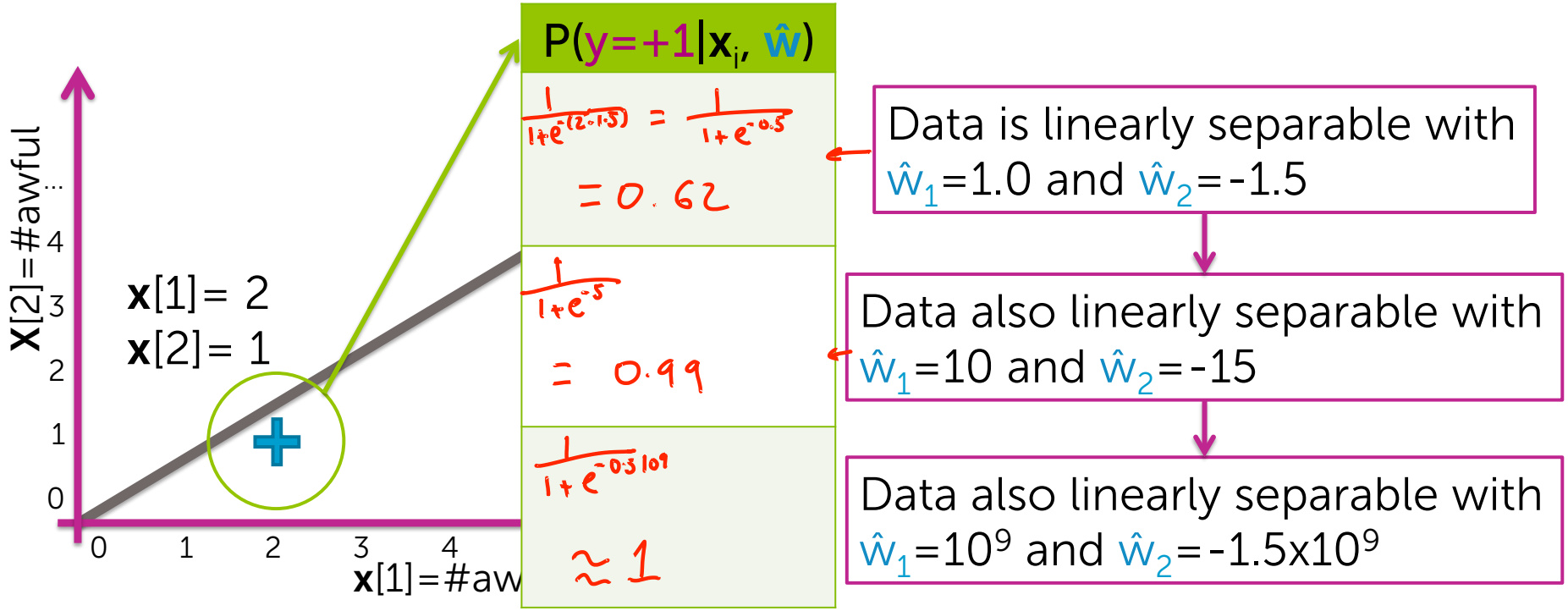


Data are linearly separable with $\hat{w}_1$=1.0 and $\hat{w}_2$=-1.5

Data also linearly separable with $\hat{w}_1$=10 and $\hat{w}_2$=-15

Data also linearly separable with $\hat{w}_1$=10^9 and $\hat{w}_2$=-1.5x10^9

Maximum likelihood estimation (MLE)
prefers most certain model ➔
Coefficients go to infinity for linearly-separable data!!!

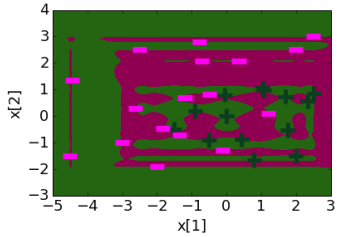


Overfitting in logistic regression is "twice as bad"

Learning tries to find decision boundary that separates data

If data are linearly separable

Overly complex boundary

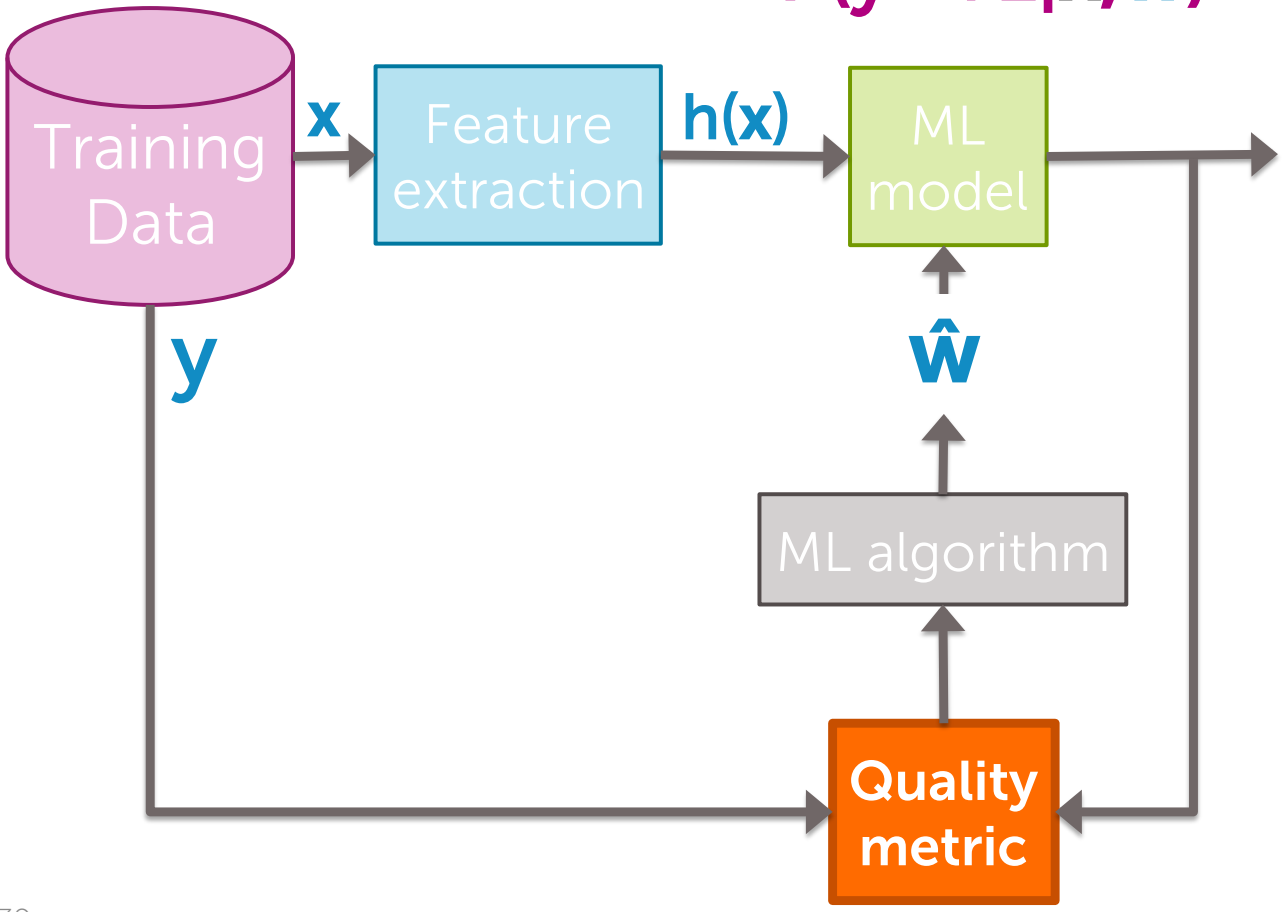


Coefficients go to infinity!

$\hat{w}_1 = 10^9$
$\hat{w}_2 = -1.5 \times 10^9$

Penalizing large coefficients to mitigate overfitting

$$\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}}) = \frac{1}{1 + e^{-\hat{\mathbf{w}}^T h(\mathbf{x})}}$$

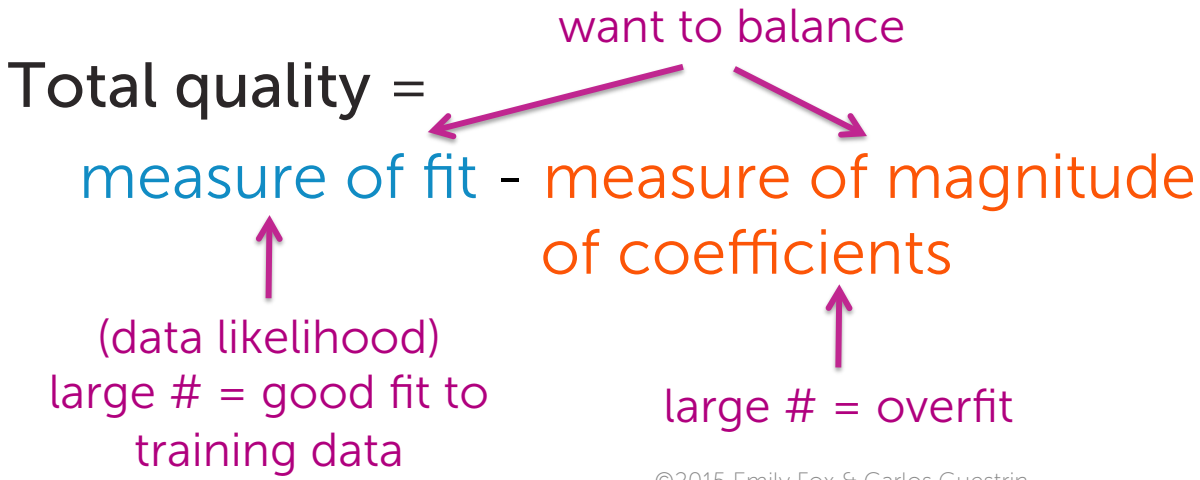


Desired total cost format

Want to balance:

i. How well function fits data

ii. Magnitude of coefficients



Maximum likelihood estimation (MLE): Measure of fit = Data likelihood

- Choose coefficients **w** that maximize likelihood:

$$\prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

- Typically, we use the log of likelihood function
  (simplifies math and has better convergence properties)

$$\ell(\mathbf{w}) = \ln \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

Measure of magnitude of logistic regression coefficients

What summary # is indicative of size of logistic regression coefficients?

- Sum of squares (L_2 norm)

$$\|w\|_2^2 = w_0^2 + w_1^2 + w_2^2 + \cdots + w_D^2$$

Penalize large coefficients

- Sum of absolute value (L_1 norm)

$$\|w\|_1 = |w_0| + |w_1| + |w_2| + \cdots + |w_D|$$

Sparse solution

Consider specific total cost

$\max_{\mathbf{w}}$

Total quality =

measure of fit - measure of magnitude of coefficients

$\ell(\mathbf{w})$ — log data likelihood

$||\mathbf{w}||_2^2$ — L_2 penalty

Consider resulting objective

What if $\hat{\mathbf{w}}$ selected to minimize

$$\ell(\mathbf{w}) - \lambda\|\mathbf{w}\|_2^2$$

λ: tuning parameter = balance of fit and magnitude

If λ=0:

→ Reduces $\max_{\mathbf{w}} \ell(\mathbf{w})$ → Standard (unpenalized) MLE solution

If λ=∞:

→ $\max_{\mathbf{w}} \ell(\mathbf{w}) - \infty \|\mathbf{w}\|_2^2$ → only care about penalizing w, large coefficients → $\mathbf{w}=0$

If λ in between:

→ Balance data fit against the magnitude of the coefficients

Consider resulting objective

What if $\hat{\mathbf{w}}$ selected to minimize

$$\ell(\mathbf{w}) - \lambda||\mathbf{w}||_2^2$$

tuning parameter = balance of fit and magnitude

L_2 regularized logistic regression

Pick λ using:
- Validation set (for large datasets)
- Cross-validation (for smaller datasets)

(see regression course)

Bias-variance tradeoff

Large λ:

high bias, low variance

(e.g., $\hat{\mathbf{w}} = 0$ for $\lambda = \infty$)

In essence, λ controls model complexity

Small λ:

low bias, high variance

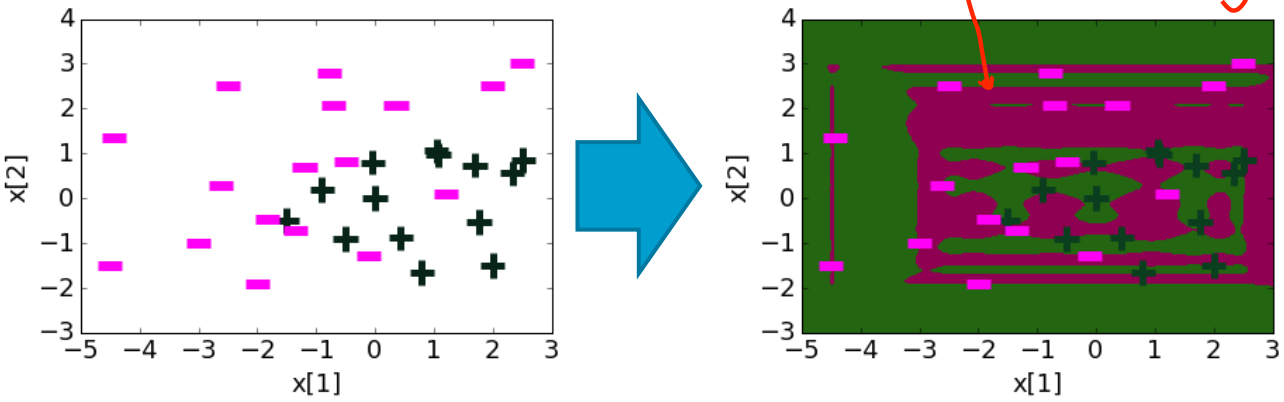
(e.g., maximum likelihood (MLE) fit of high-order polynomial for $\lambda = 0$)

Visualizing effect of regularization on logistic regression

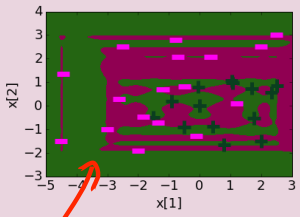
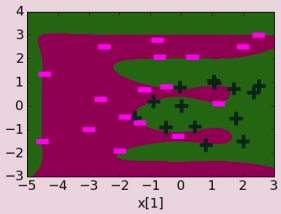
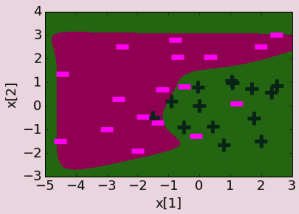
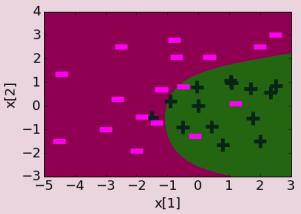
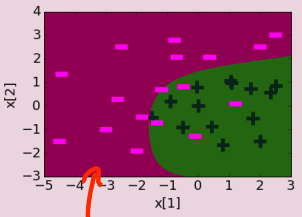
Degree 20 features, λ=0

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	8.7
$h_1(\mathbf{x})$	$\mathbf{x}[1]$	5.1
$h_2(\mathbf{x})$	$\mathbf{x}[2]$	78.7
…	…	…
$h_{11}(\mathbf{x})$	$(\mathbf{x}[1])^6$	-7.5
$h_{12}(\mathbf{x})$	$(\mathbf{x}[2])^6$	3803
$h_{13}(\mathbf{x})$	$(\mathbf{x}[1])^7$	21.1
$h_{14}(\mathbf{x})$	$(\mathbf{x}[2])^7$	-2406
…	…	…
$h_{37}(\mathbf{x})$	$(\mathbf{x}[1])^{19}$	$-2*10^{-6}$
$h_{38}(\mathbf{x})$	$(\mathbf{x}[2])^{19}$	-0.15
$h_{39}(\mathbf{x})$	$(\mathbf{x}[1])^{20}$	$-2*10^{-8}$
$h_{40}(\mathbf{x})$	$(\mathbf{x}[2])^{20}$	0.03

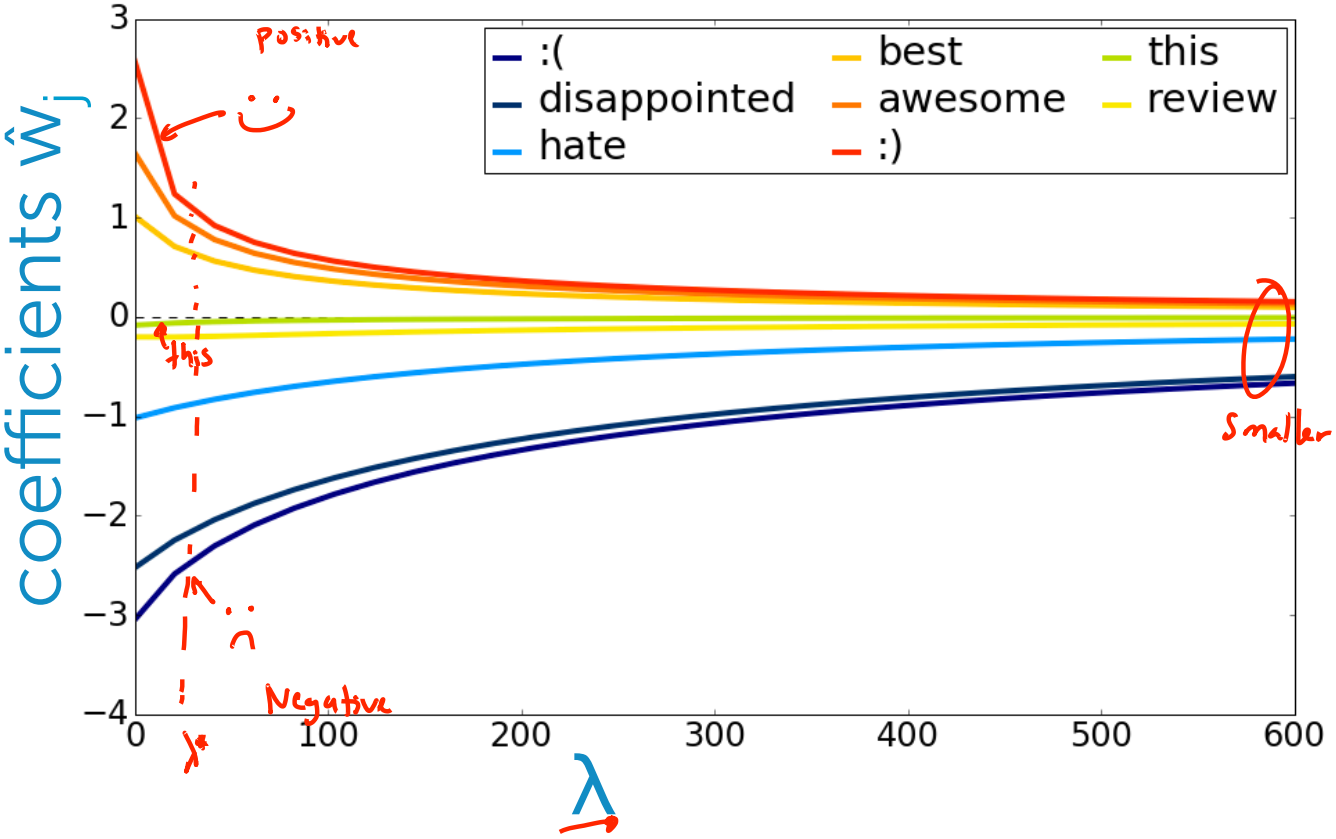
Coefficients range from **-3170** to **3803**



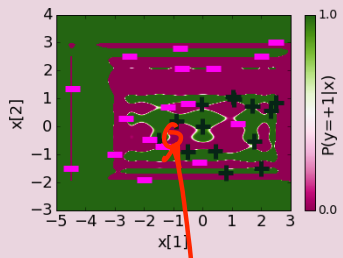
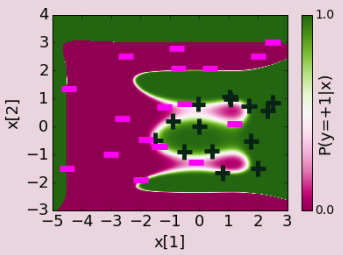
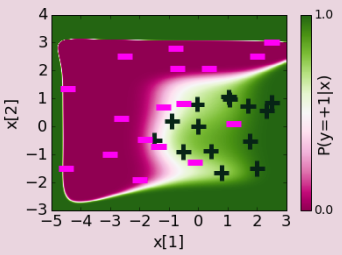
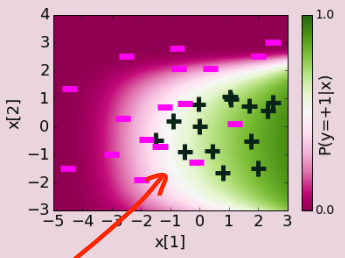
Degree 20 features, effect of regularization penalty λ

Regularization	λ = 0	λ = 0.00001	λ = 0.001	λ = 1	λ = 10
Range of coefficients	-3170 to 3803	-8.04 to 12.14	-0.70 to 1.25	-0.13 to 0.57	-0.05 to 0.22
Decision boundary					

Coefficient path



Degree 20 features: regularization reduces "overconfidence"

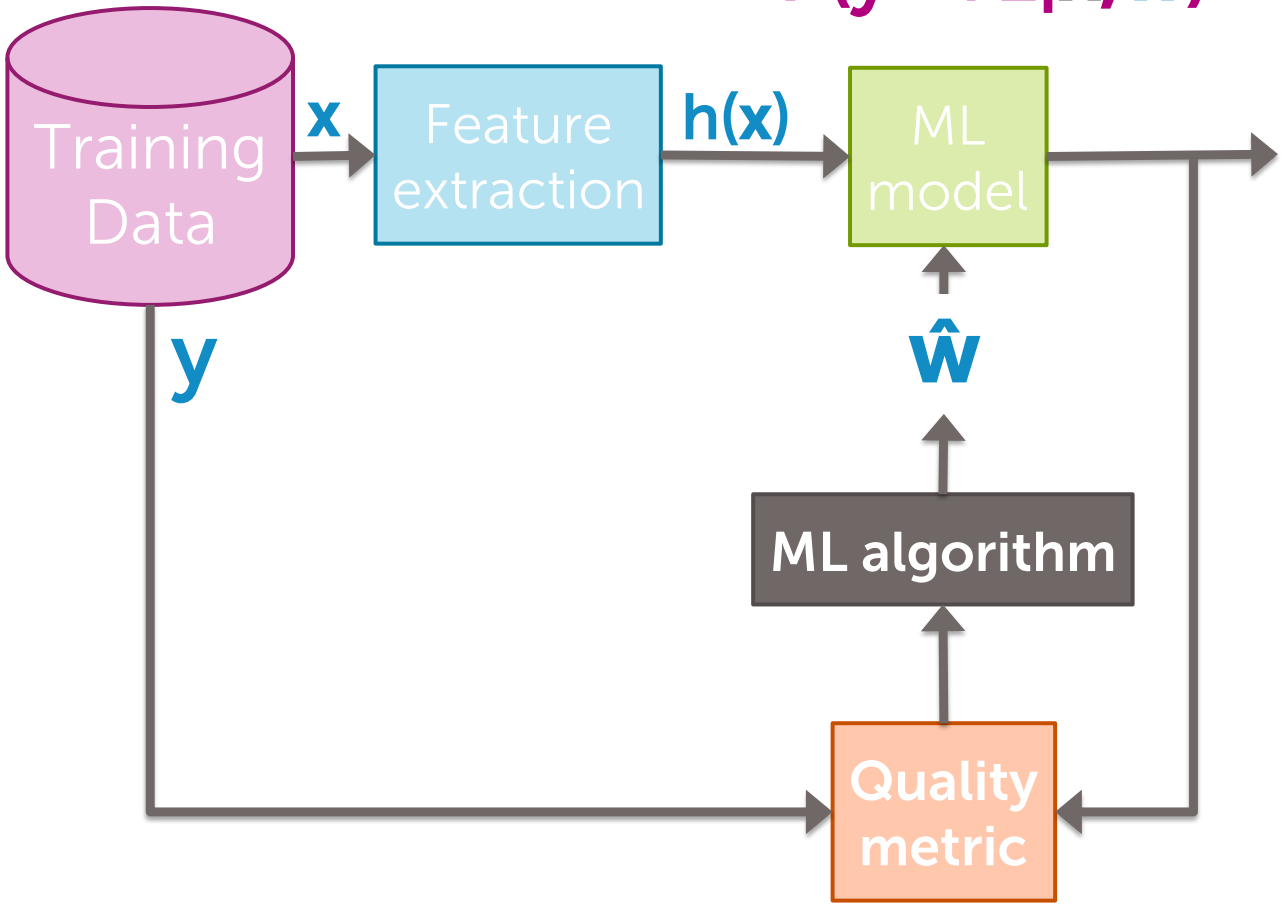
Regularization	λ = 0	λ = 0.00001	λ = 0.001	λ = 1
Range of coefficients	-3170 to 3803	-8.04 to 12.14	-0.70 to 1.25	-0.13 to 0.57
Learned probabilities				

highly over confident

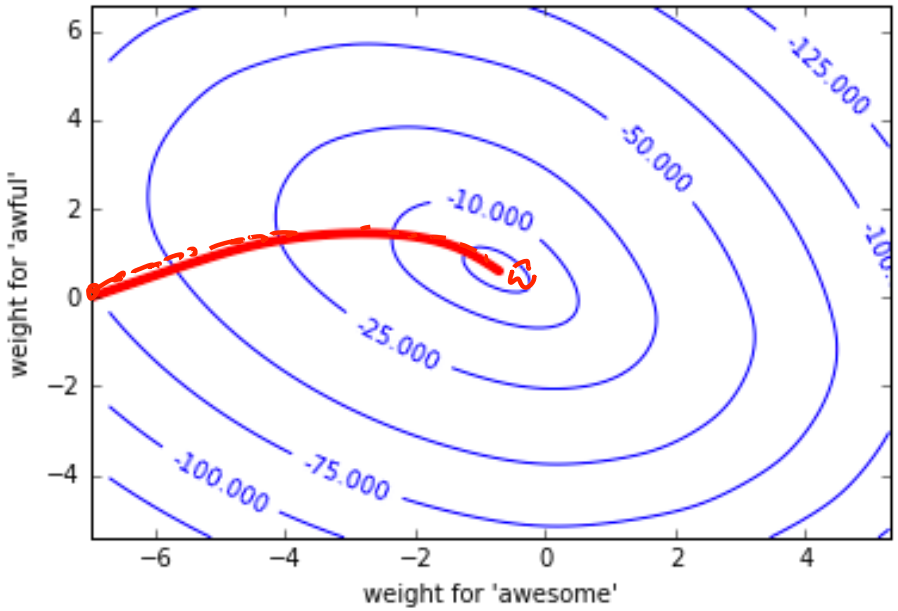
very natural uncertainty region

Finding best L_2 regularized linear classifier with gradient ascent

$$\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}}) = \frac{1}{1 + e^{-\hat{\mathbf{w}}^T h(\mathbf{x})}}$$



Gradient ascent



Algorithm:

while not converged

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta \nabla \ell(\mathbf{w}^{(t)})$$

need the gradient of regularized log likelihood

Gradient of L_2 regularized log-likelihood

Total quality =

measure of fit - measure of magnitude of coefficients

$\ell(\mathbf{w})$ $\qquad$ $\lambda \|\mathbf{w}\|_2^2$

Total derivative = $\dfrac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} - \lambda \dfrac{\partial \|\mathbf{w}\|_2^2}{\partial \mathbf{w}_j}$

Derivative of (log-)likelihood

Sum over data points

Feature value

Difference between truth and prediction

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w})\Big)$$

Derivative of L_2 penalty

$$\frac{\partial ||\mathbf{w}||_2^2}{\partial \mathbf{w}_j} = \frac{\partial}{\partial w_j}\left[w_0^2 + w_1^2 + w_2^2 + \cdots + w_j^2 + \cdots + w_D^2\right] = 2w_j$$

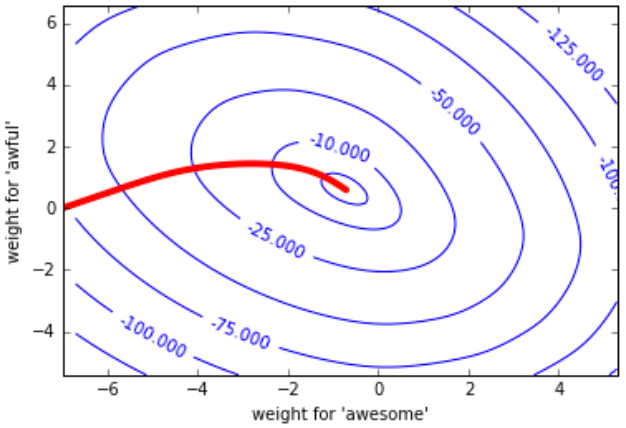
Understanding contribution of L_2 regularization

Term from L_2 penalty

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \overbrace{- 2\lambda \mathbf{w}_j}$$

	$-2\lambda w_j$	Impact on w_j
$w_j > 0$	< 0	decreases w_j => w_j becomes closer to 0
$w_j < 0$	> 0	increases w_j => w_j becomes closer to 0

Summary of gradient ascent for logistic regression with L_2 Regularization



init $\mathbf{w}^{(1)}=0$ (or randomly, or smartly), t=1

while not converged:

for j=0,…,D

$$\text{partial[j]} = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\right)$$

$$\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} + \eta\,(\text{partial[j]} - 2\lambda\,\mathbf{w}_j^{(t)})$$

$t \leftarrow t + 1$

step size

$\frac{\partial \ell(w)}{\partial w_j}$

only change w_j

Sparse logistic regression with L_1 regularization

Recall sparsity (many $\hat{w}_j$=0) gives efficiency and interpretability

Efficiency:

- If size($\mathbf{w}$) = 100B, each prediction is expensive
- If $\hat{\mathbf{w}}$ sparse, computation only depends on # of non-zeros

many zeros

$$\hat{y}_i = sign\left(\sum_{\hat{\mathbf{w}}_j \neq 0} \hat{\mathbf{w}}_j h_j(\mathbf{x}_i)\right)$$

Interpretability:

- Which features are relevant for prediction?

Sparse logistic regression

Total quality =

measure of fit - measure of magnitude of coefficients

$\ell(\mathbf{w})$ $\quad ||\mathbf{w}||_1 = |w_0| + \ldots + |w_D|$

Leads to **sparse** solutions!

L_1 regularized logistic regression

L_1 regularized logistic regression

Just like L2 regularization, solution is governed by a continuous parameter λ

$$\ell(\mathbf{w}) - \lambda||\mathbf{w}||_1$$

tuning parameter = balance of fit and **sparsity**

If λ=0:

No regularization → standard MLE solution

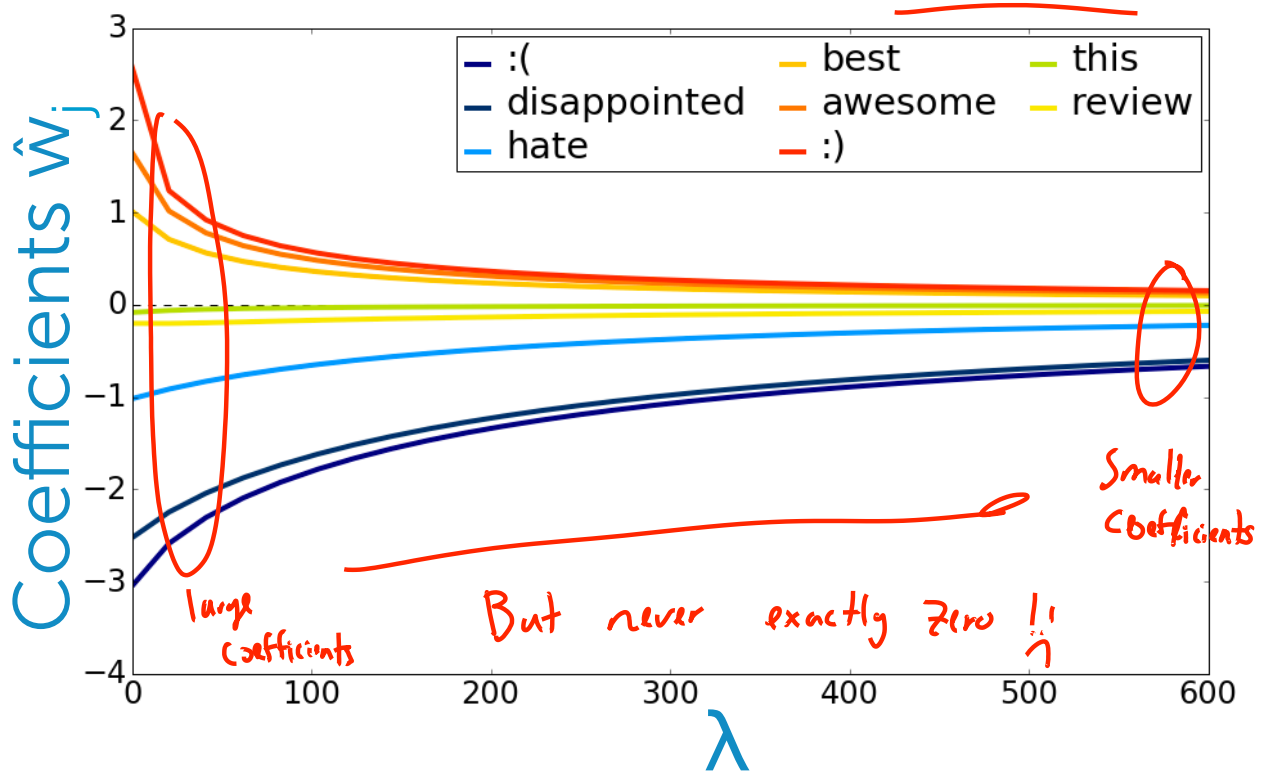
If λ=∞:

all weight is on regularization → $\hat{w} = 0$

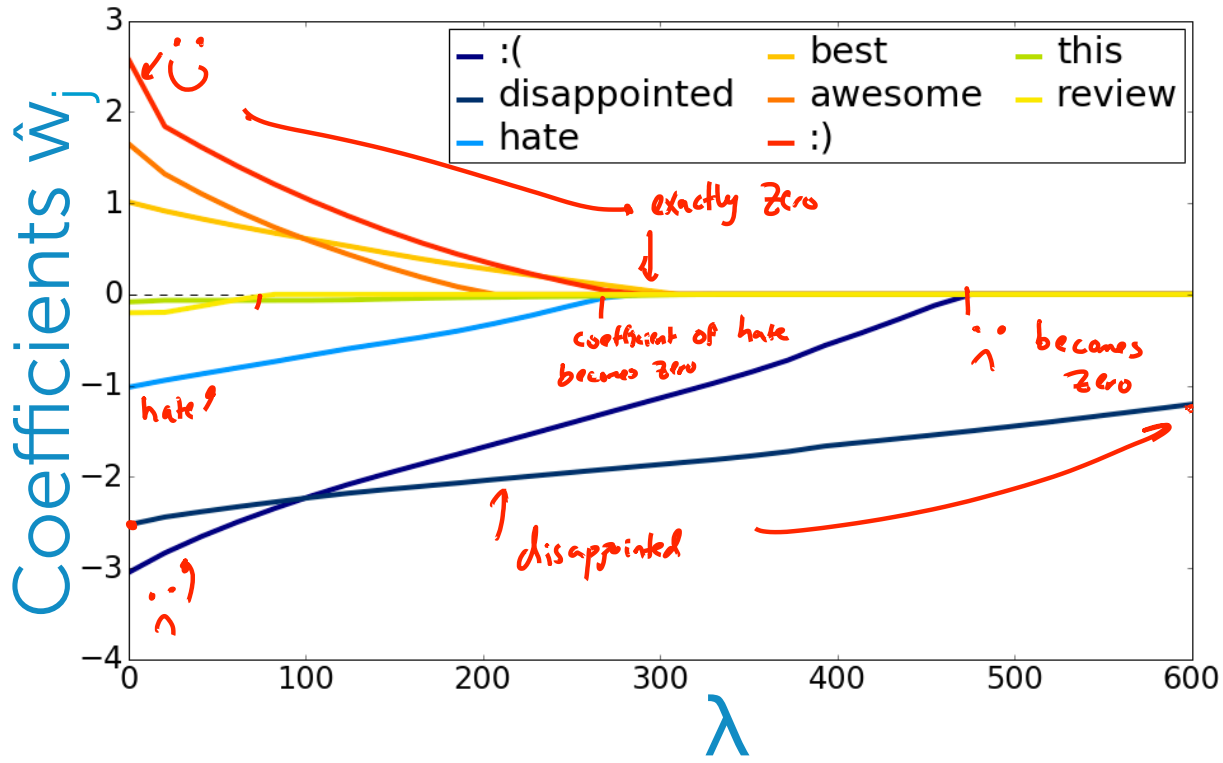
If λ in between:

Sparse solutions: Some $\hat{w}_j \neq 0$, many other $\hat{w}_j = 0$

Regularization path – L_2 penalty



Regularization path – L_1 penalty



Summary of overfitting in logistic regression

What you can do now…

- Identify when overfitting is happening
- Relate large learned coefficients to overfitting
- Describe the impact of overfitting on decision boundaries and predicted probabilities of linear classifiers
- Motivate the form of L_2 regularized logistic regression quality metric
- Describe what happens to estimated coefficients as tuning parameter λ is varied
- Interpret coefficient path plot
- Estimate L_2 regularized logistic regression coefficients using gradient ascent
- Describe the use of L_1 regularization to obtain sparse logistic regression solutions