Logistic Regression

Machine Learning

Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Models and learning algorithms

- Last week, we saw how to decouple model from learning algorithm
- (Stochastic) Gradient Descent can train various models
- Today, a classification model
  - can be fit using (S)GD

Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Error function

- Learning a model – minimizing error function
- Different models – different error functions
- For example, SSE for linear regression

$$\mathrm{SSE} = \sum_{i=1}^{N} (y^i_{\mathrm{pred}} - y^i)^2$$

Loss function

- Commonly formulated as quantifying our mistake on a **single example**
- **Squared loss** corresponds to **SSE**.

$$\ell_{\text{squared}}(z) = (z - y)^2$$

where $z = \mathbf{w} \cdot \mathbf{x} + b$ is the prediction of the model and y the target value

Agenda

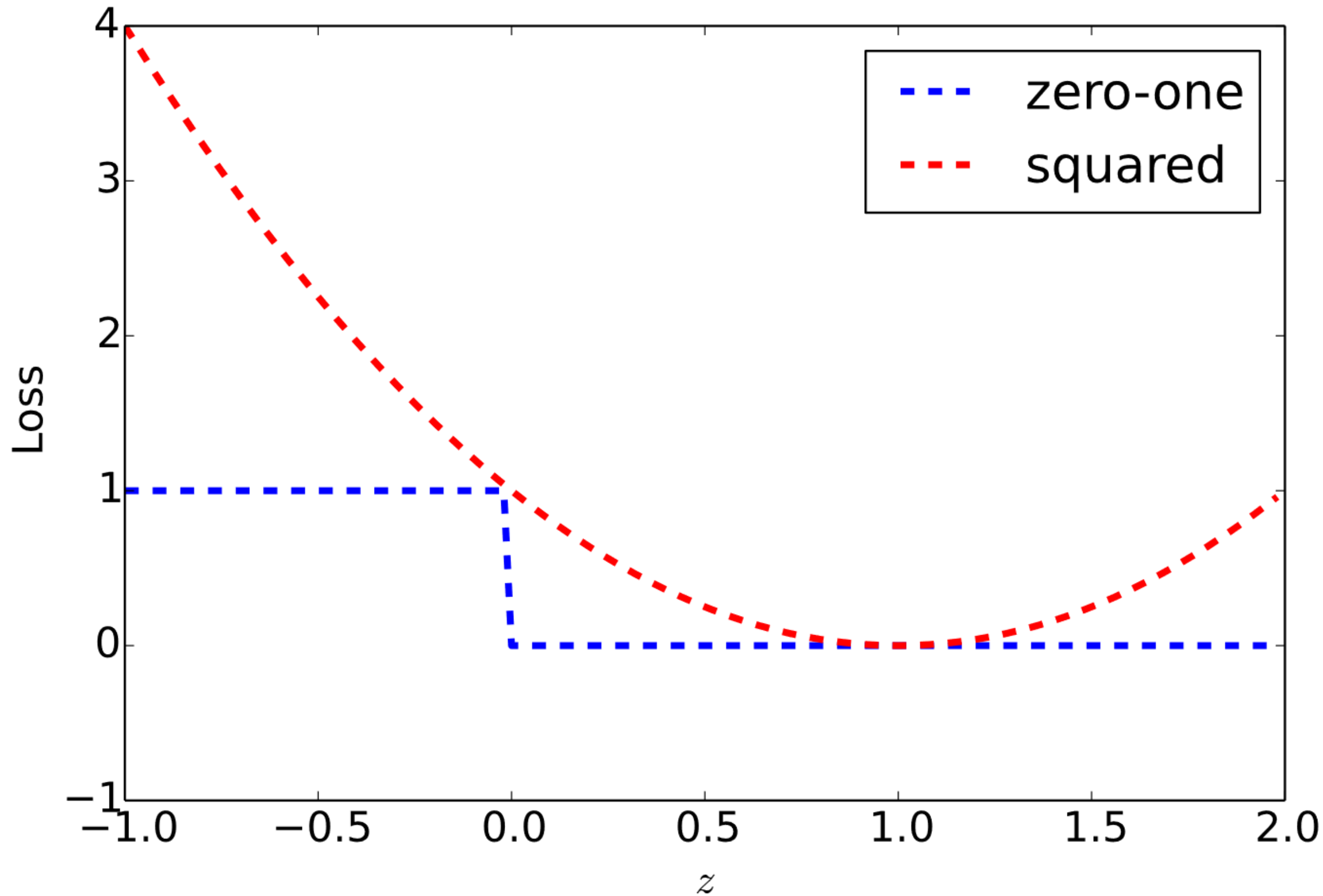
- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Loss for classification

Zero-one loss:

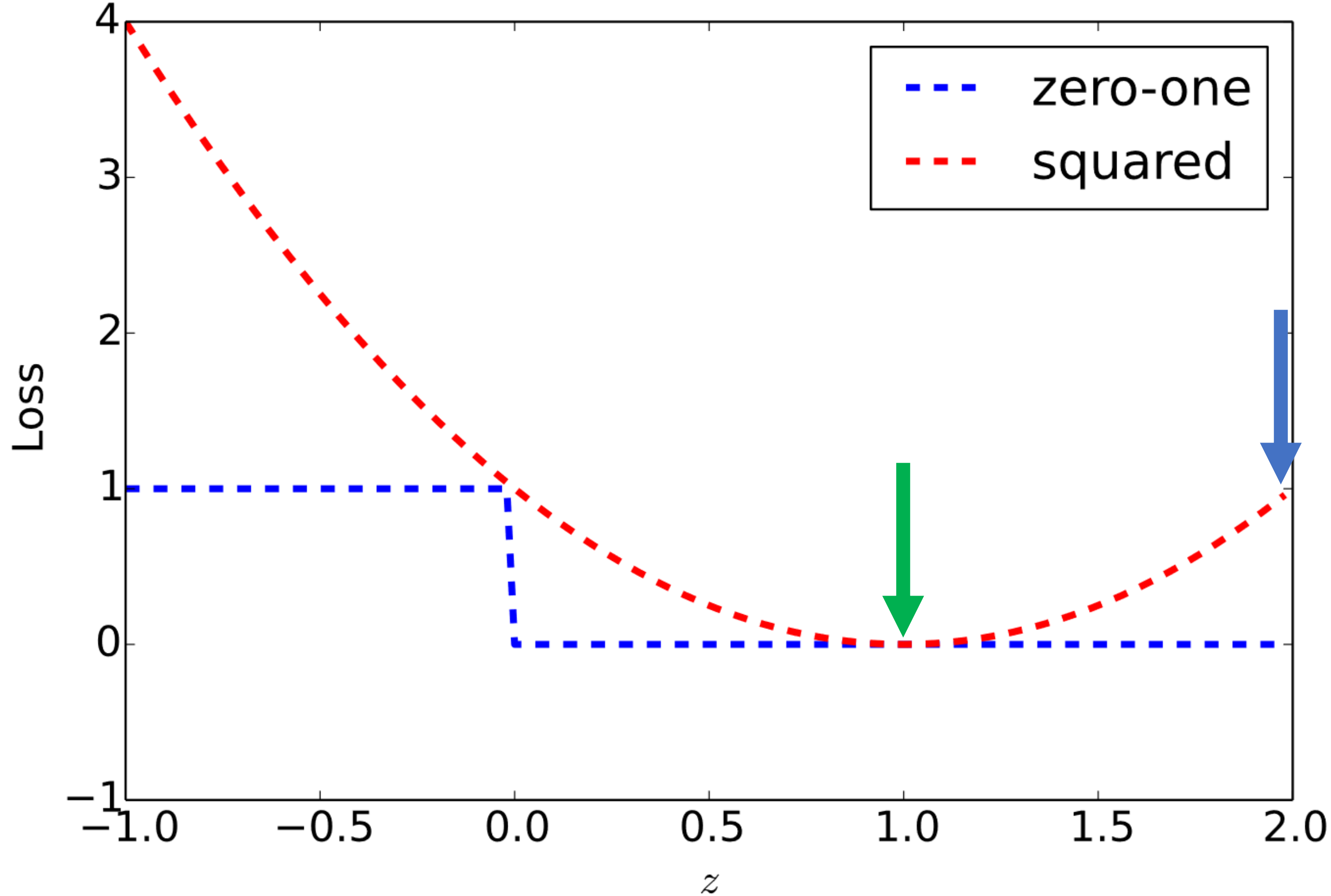
1 if we've made a mistake, 0 otherwise

$$\ell_{0/1}(z) = \begin{cases} 1 \text{ if } y = 1 \text{ and } z < 0 \\ 1 \text{ if } y = -1 \text{ and } z >= 0 \\ 0 \text{ otherwise} \end{cases}$$


zero-one
squared
Loss
z
4
3
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Loss for classification

- Zero-one loss and gradient descent
  - Gradient is zero, not useful
- Could we just use squared loss for classification?
  - Treat $z = 1.0$ as the correct label for $y = +1$

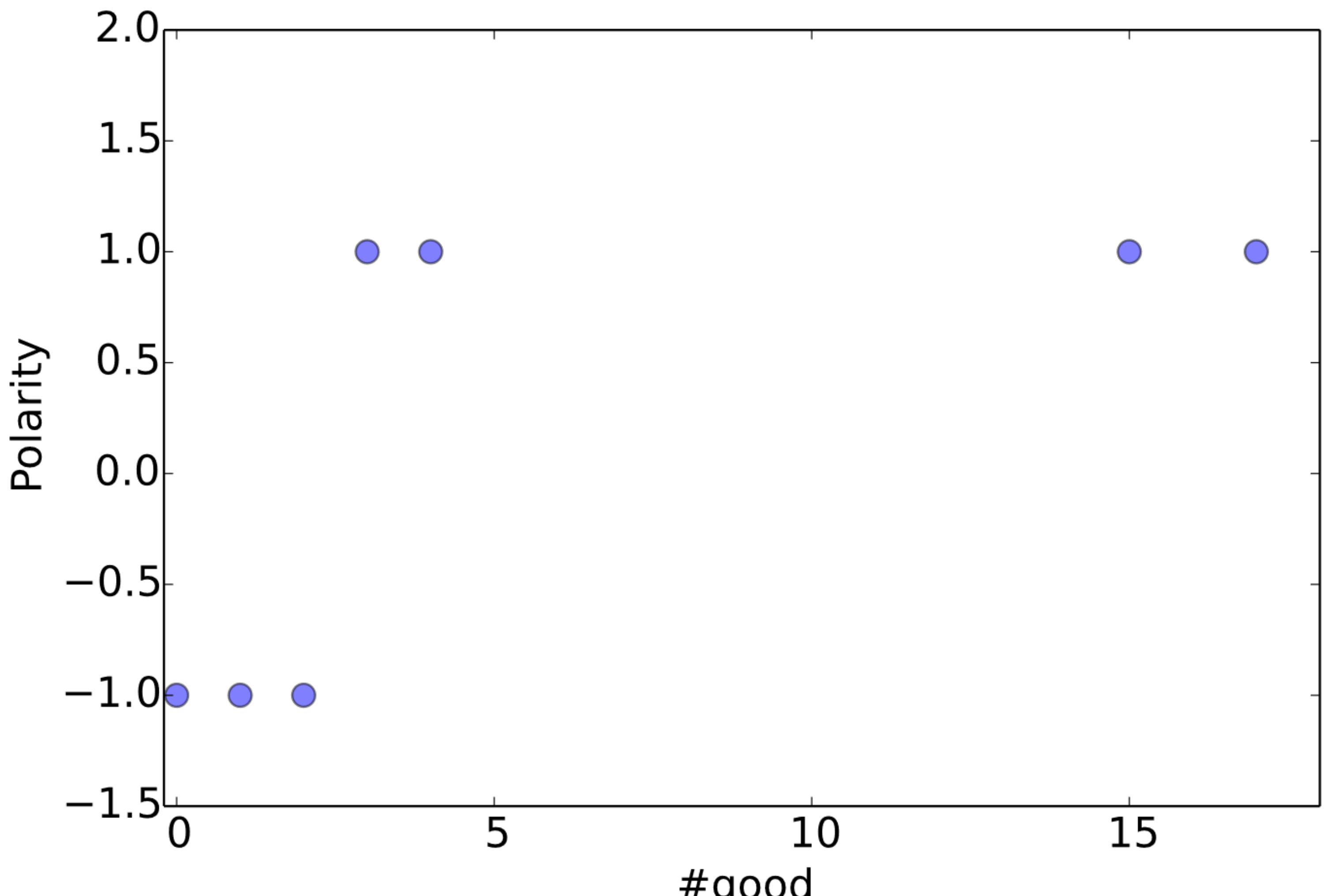


zero-one
squared
Loss
4
3
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
z

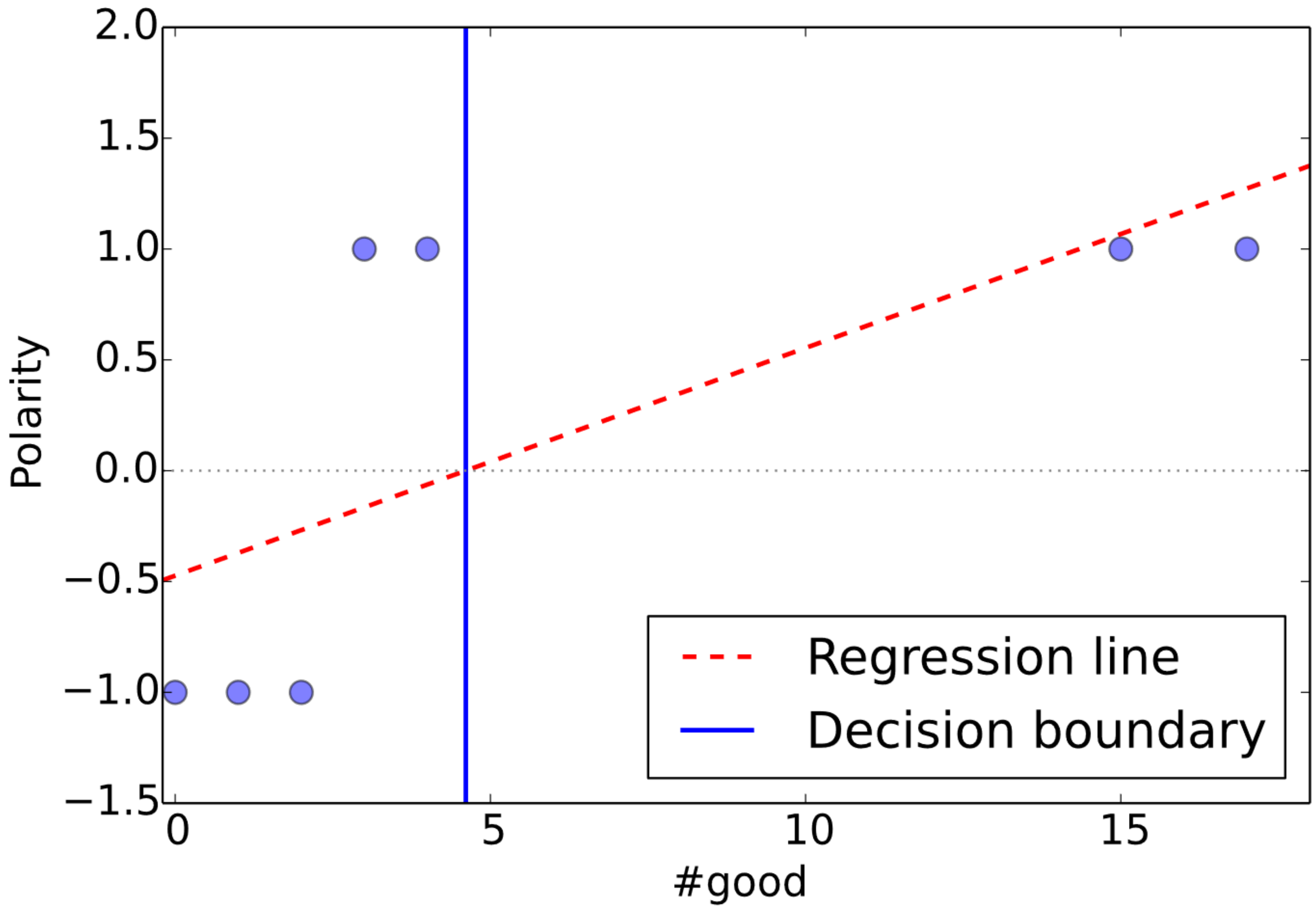
Loss for classification

- Zero-one loss and gradient descent
  - Gradient is zero, not useful
- Could we just use squared loss for classification?
  - No, penalizes confident correct predictions

Example



Example: Squared Loss



Problem

- Bad decision boundary
- Model cares too much about predicting exactly 1 for examples with high *#good*
- Need better loss function

Regression for classifying

- In regression we predict numbers
- In classification we predict labels
- Regress on **probabilities** of labels
  - This is logistic regression!

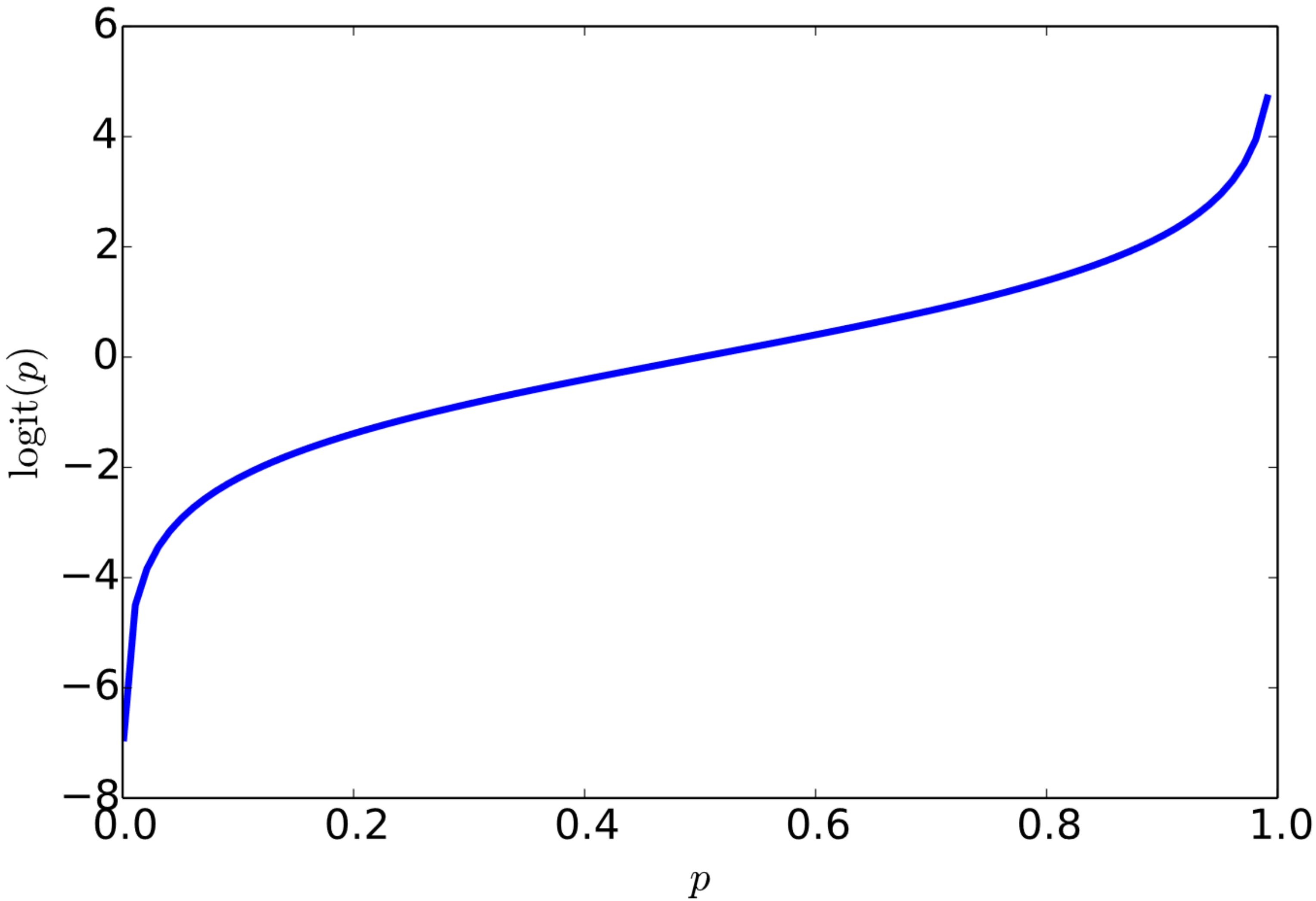
Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

The Logit function

- Let p = probability that label is positive
    - number between 0 and 1
- Logit function maps p to [-∞,∞]

$$\operatorname{logit}(p) = \log\left(\frac{p}{1-p}\right)$$



Examples

logit(0.01) = -4.6
logit(0.50) = 0.0
logit(0.99) = 4.6

Logistic regression : The regression part

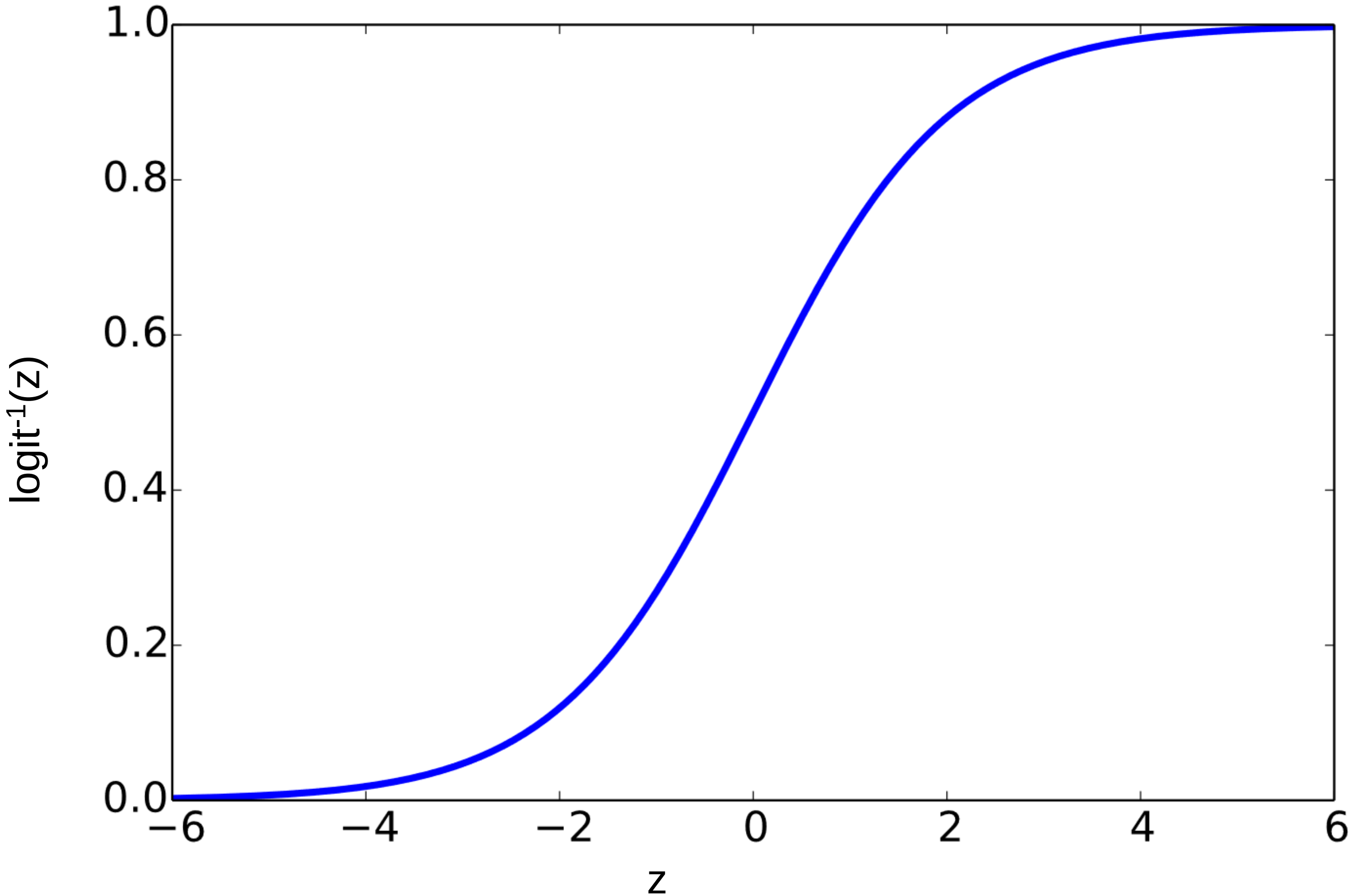
A logistic regression model uses a linear model to predict logit(p)

$$\operatorname{logit}(p)_{\text{pred}} = \mathbf{w} \cdot \mathbf{x} + b$$

Logistic regression: The probability part

We can map the logit back to probability using the **inverse logit** (or **logistic,** or **sigmoid**) function

$$\text{logit}^{-1}(z) = \frac{1}{1 + \exp(-z)}$$



Examples

$\text{logit}^{-1}(0) = 0.50$
$\text{logit}^{-1}(-4.6) = 0.01$
$\text{logit}^{-1}(4.6) = 0.99$

Logistic regression

Putting the pieces together

$$p_{\mathrm{pred}} = \mathrm{logit}^{-1}(\mathbf{w} \cdot \mathbf{x} + b)$$

Example: movie reviews

	#good	#dark	#mediocre	#the
$\mathbf{x}^1$ = (	1,	0,	0,	5)
$\mathbf{x}^2$ = (	2,	3,	2,	7)
$\mathbf{w}$ = (	2.5,	0.5,	-4.0,	0.0)

b = 0.5

$\text{score}^1 = 3.0$, $p^1 = \text{logit}^{-1}(3.0) = 0.95$

$\text{score}^2 = -1.0$, $p^2 = \text{logit}^{-1}(-1.0) = 0.27$

Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Reminder: Logistic regression definition

$$p_{\text{pred}} = \text{logit}^{-1}(\mathbf{w} \cdot \mathbf{x} + b)$$

Log loss function aka cross-entropy

Loss function quantifying mistakes for LR

$$\ell_{\log}(z) = \begin{cases} -\log(p_{\text{pred}}) & \text{if } y = 1 \\ -\log(1 - p_{\text{pred}}) & \text{if } y = 0 \end{cases}$$

$$\text{where } p_{\text{pred}} = \text{logit}^{-1}(z)$$

Minimize log loss – find model which gives **maximum probability** to training targets

Log loss function: a common alternative formulation

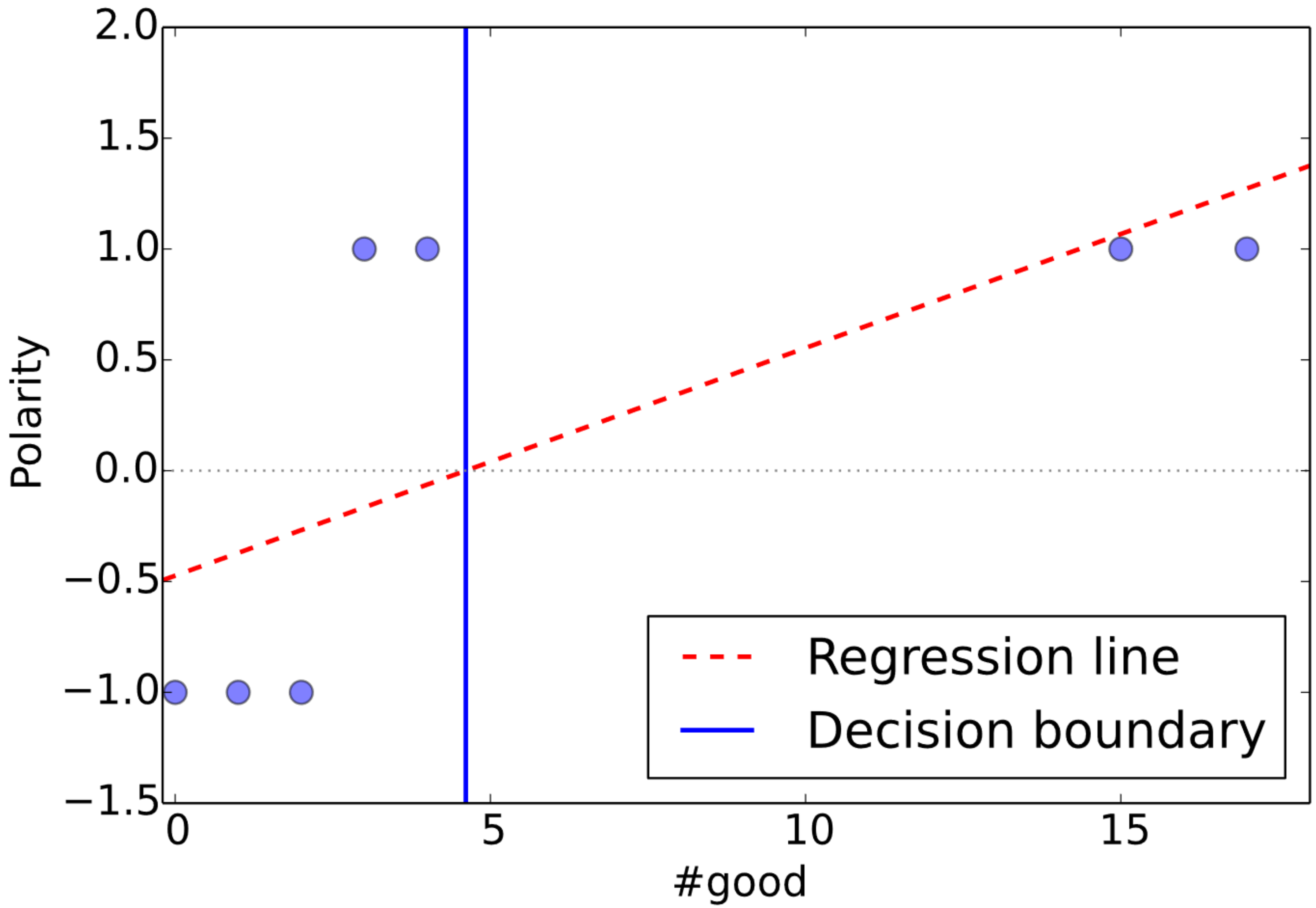
$$\ell_{\log}(z) = \begin{cases} -\log(p_{\text{pred}}) & \text{if } y = 1 \\ -\log(1 - p_{\text{pred}}) & \text{if } y = 0 \end{cases}$$

$$\text{where } p_{\text{pred}} = \text{logit}^{-1}(z)$$

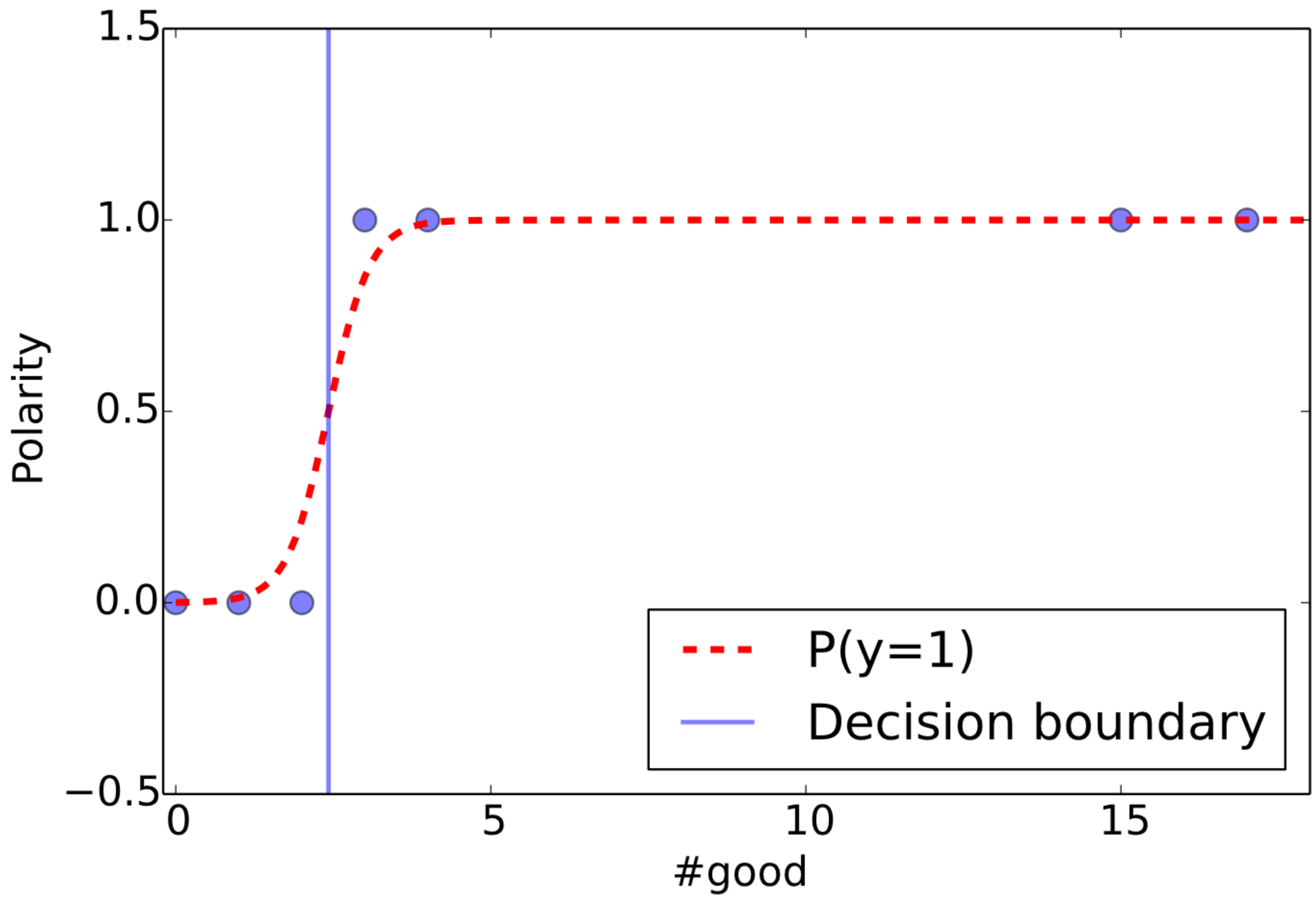
alternative notation

$$\ell_{\log}(z) = -y \log(p_{\text{pred}}) - (1 - y) \log(1 - p_{\text{pred}})$$

Example: Squared Loss



Example: Log loss



Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Logistic vs Linear: prediction

- Both use the score of the linear model

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

- Linear regression uses it directly

$$y_{\text{pred}} = z$$

- Logistic regression via inverse logit

$$p_{\text{pred}} = \text{logit}^{-1}(z)$$

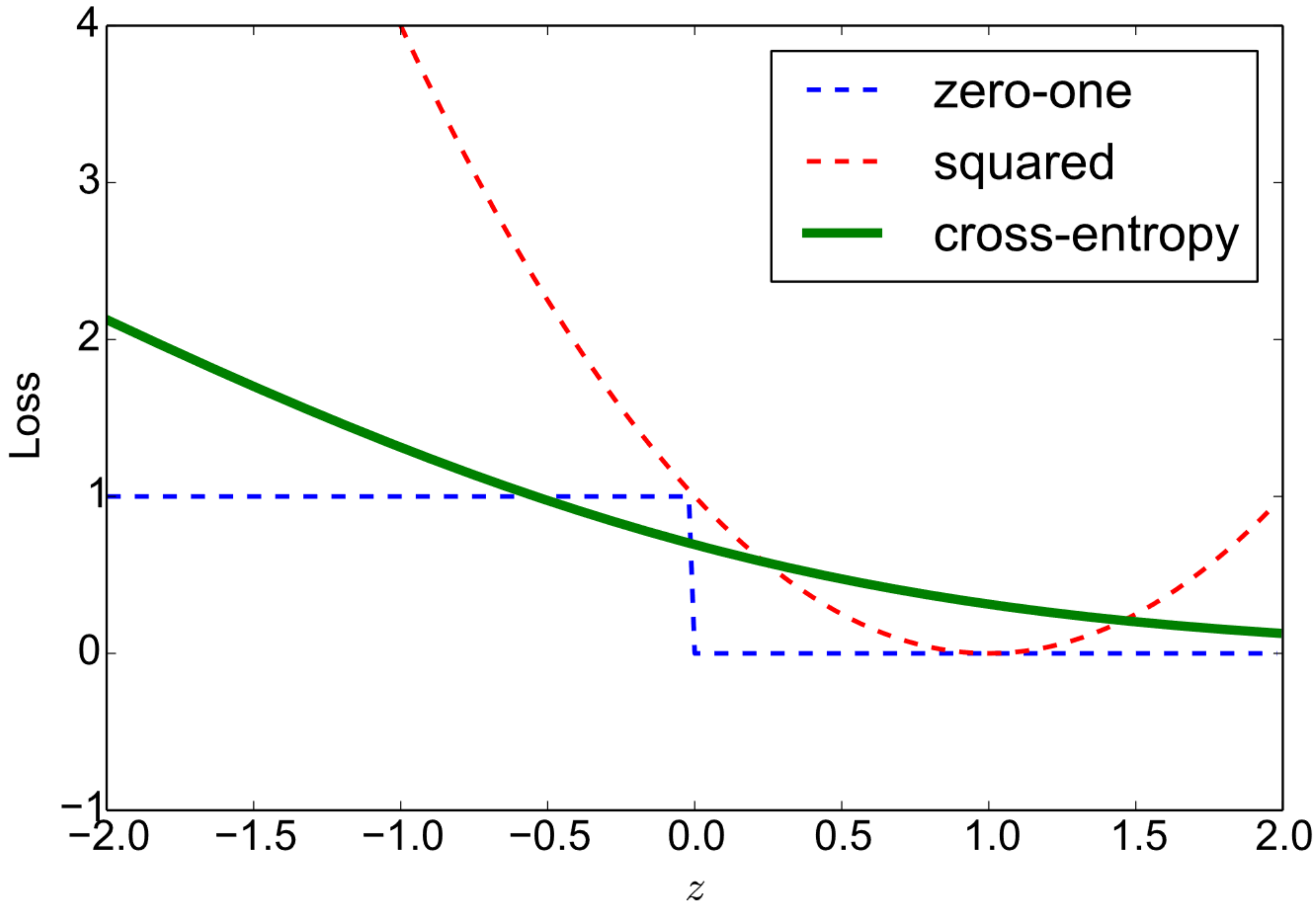
Logistic vs Linear: loss

- For linear regression use squared loss

$$\ell_{\text{squared}}(z) = (z - y)^2$$

- For logistic regression use cross-entropy

$$\ell_{\log}(z) = -y \log(p_{\text{pred}}) - (1 - y) \log(1 - p_{\text{pred}})$$


zero-one
squared
cross-entropy
Loss
z
4
3
2
1
0
−1
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Logistic vs Linear: SGD

- Both models can be learned via **Stochastic Gradient Descent**
- Linear regression

$$\mathrm{w}_{\mathrm{new}} = \mathbf{w}_{\mathrm{old}} - \eta \times 2(y_{\mathrm{pred}} - y)\mathbf{x}$$

- Logistic regression

$$\mathrm{w}_{\mathrm{new}} = \mathbf{w}_{\mathrm{old}} + \eta \times (y - p_{\mathrm{pred}})\mathbf{x}$$

Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Logistic vs Perceptron: prediction

- Both use the score of the linear model

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

- Perceptron passes it through a threshold function

$$y_{\text{pred}} = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Logistic regression through inverse logit

$$p_{\text{pred}} = \text{logit}^{-1}(z)$$

Perceptron vs LR update

- Perceptron

$$\mathrm{w}_{\mathrm{new}} = \mathbf{w}_{\mathrm{old}} \pm \mathbf{x}$$ (Depending on direction of error)

- Logistic regression

$$\mathrm{w}_{\mathrm{new}} = \mathbf{w}_{\mathrm{old}} + \eta \times \left(y - p_{\mathrm{pred}}\right)\mathbf{x}$$

Agenda

- Review and framing this week
- Error functions and loss functions
- Possible error functions for Classification
- The logit function and it's inverse
- The Log loss function
- Logistic vs Linear regression
- Logistic regression vs Perceptron
- Logistic regression and weights

Impact of restricting weights

- Models with small, less variable weights are less flexible
  - Less freedom to fit data
- Penalize weight variance can help reduce overfitting

L2 regularization penalty

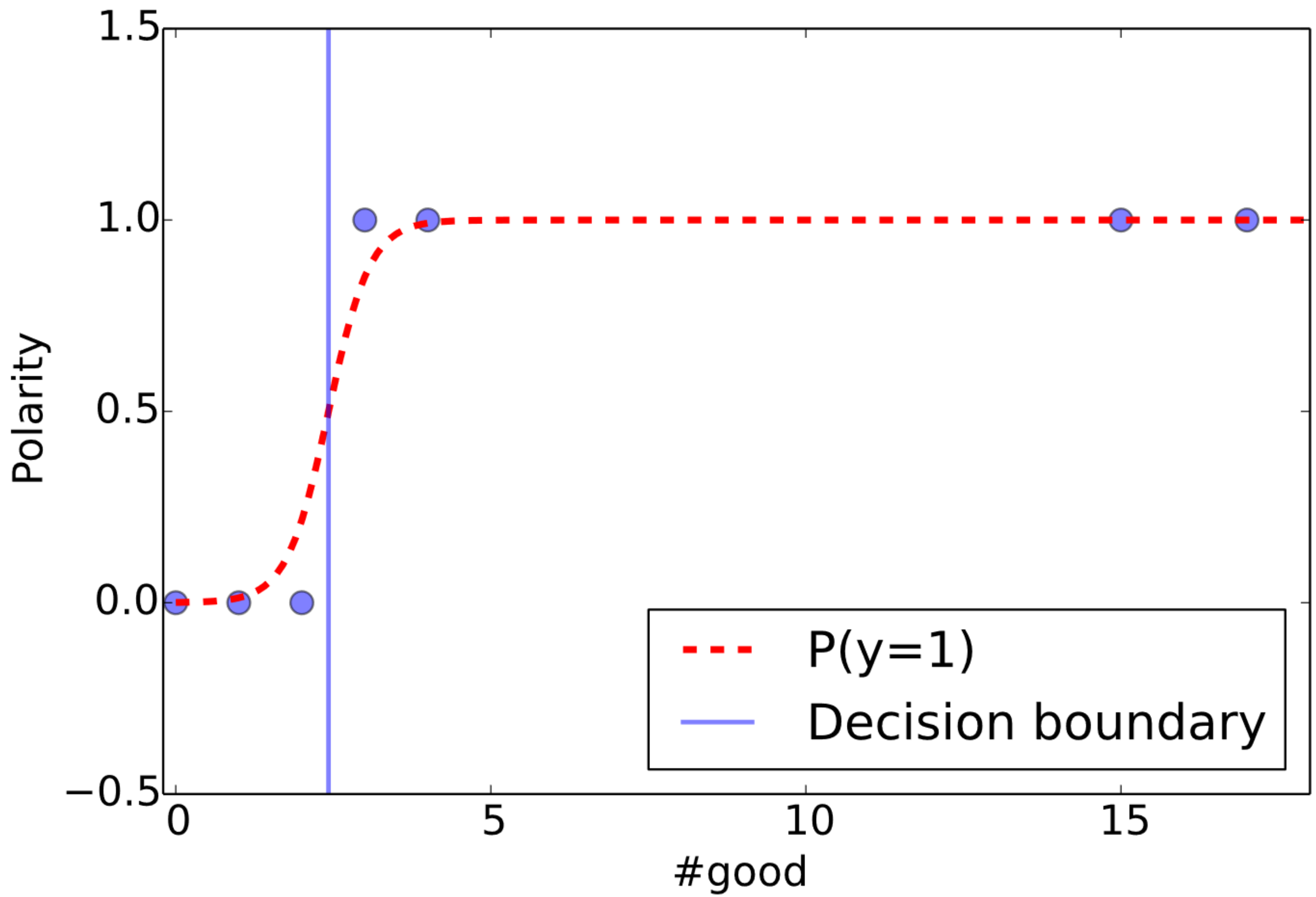
$$\mathrm{Error}(\mathbf{w}, b) = \sum_{i=1}^{N} \ell(z^i) + \alpha \sum_{j=1}^{M} w_j^2$$

make model fit

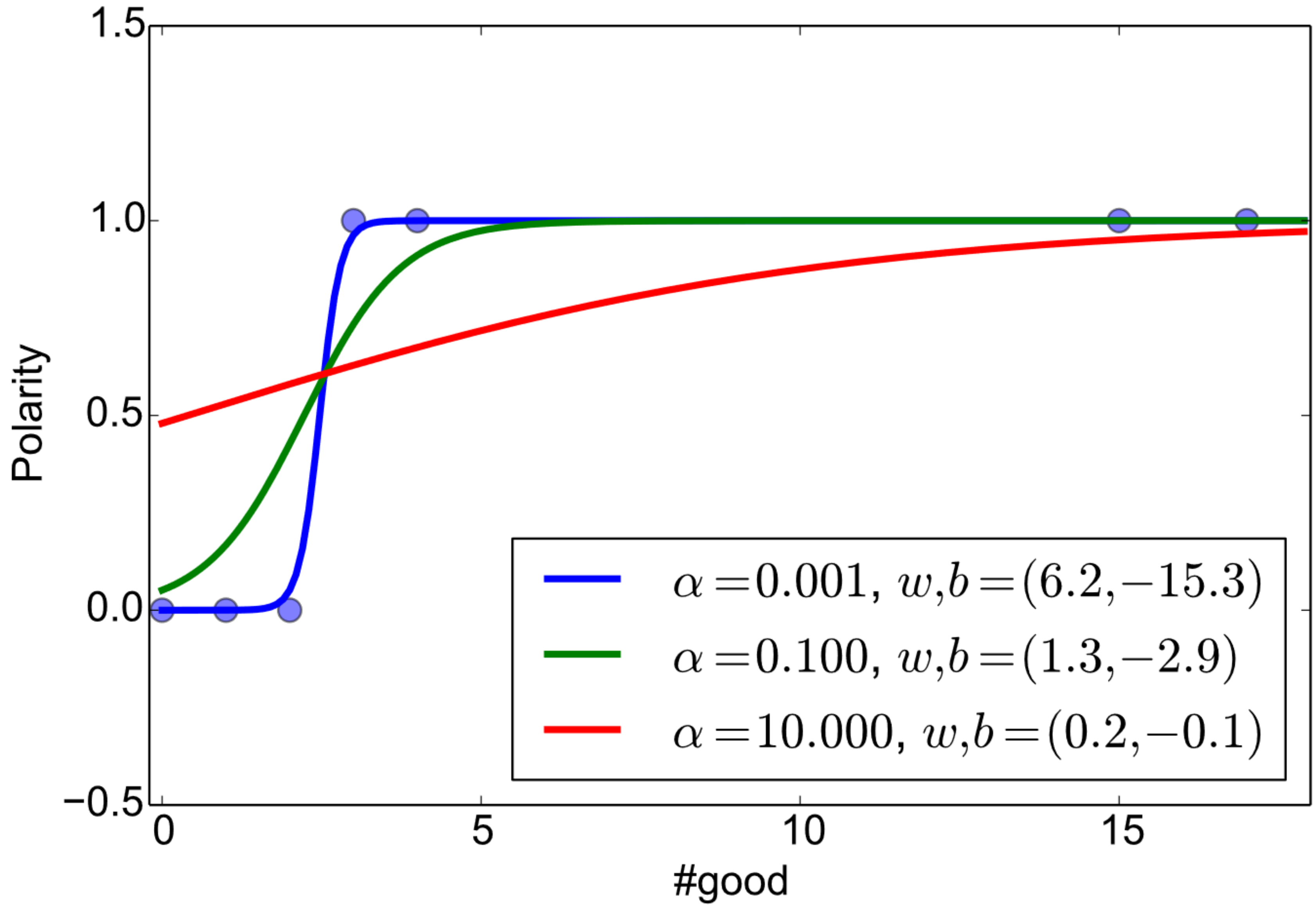
trade-off

make model “simple”

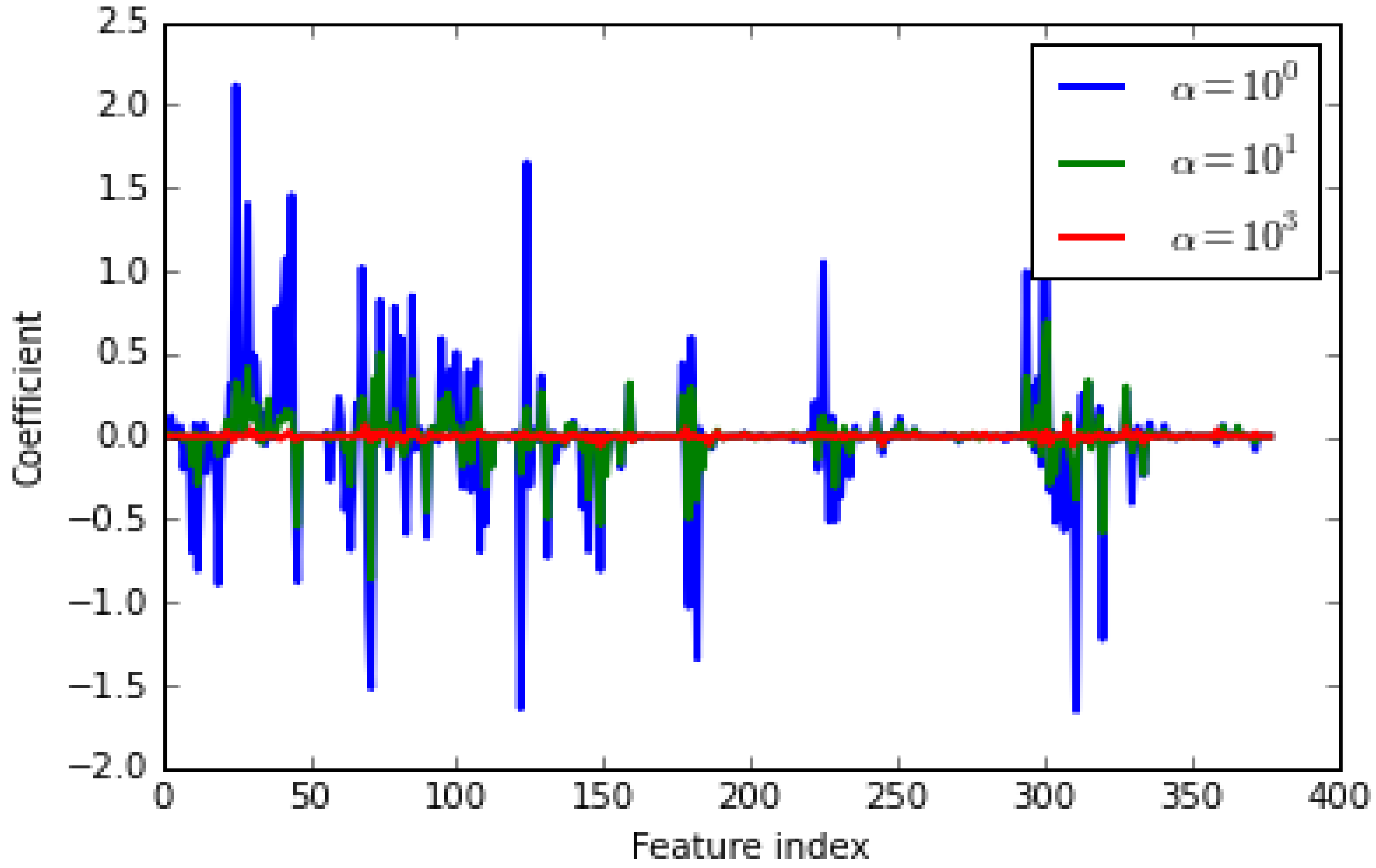
Example: Log loss



Example: Regularization



Alpha and Weight variance



Logistic regression in scikit-learn

- `sklearn.linear_model.SGDClassifier`
  - Supports several loss functions including log loss
  - Good choice for large datasets
  - Regularization via parameter **alpha**
- `sklearn.linear_model.LogisticRegression`
  - Specialized LR algorithm
  - Good for small and medium data sizes
  - Regularization via param **C=1/alpha**

Summary

- Different loss functions give rise to different linear models
- Logistic regression for probabilistic classification
- Control overfitting via L2 regularization