10: Empirical Risk Minimization

Cornell CS 4/5780

Recap

Remember the unconstrained SVM Formulation

$$\min_{\mathbf{w}} \; \underbrace{C\sum_{i=1}^{n} \max[1 - y_i \underbrace{(w^\top \mathbf{x}_i + b)}_{h(\mathbf{x}_i)}, 0]}_{Hinge-Loss} + \underbrace{\| w \|_2^2}_{l_2-Regularizer}$$

The hinge loss is the SVM's error function of choice, whereas the l_2-regularizer penalizes (overly) complex solutions. This is an example of empirical risk minimization with a loss function ℓ and a regularizer r,

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\ell(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{Loss} + \underbrace{\lambda r(w)}_{Regularizer},$$

where the loss function is a continuous function which penalizes training error, and the regularizer is a continuous function which penalizes classifier complexity. Here, we define λ as $\frac{1}{C}$ from the previous lecture.[1]

Commonly Used Binary Classification Loss Functions

Different Machine Learning algorithms use different loss functions; Table 4.1 shows just a few (here we assume $y_i \in \{+1, -1\}$):

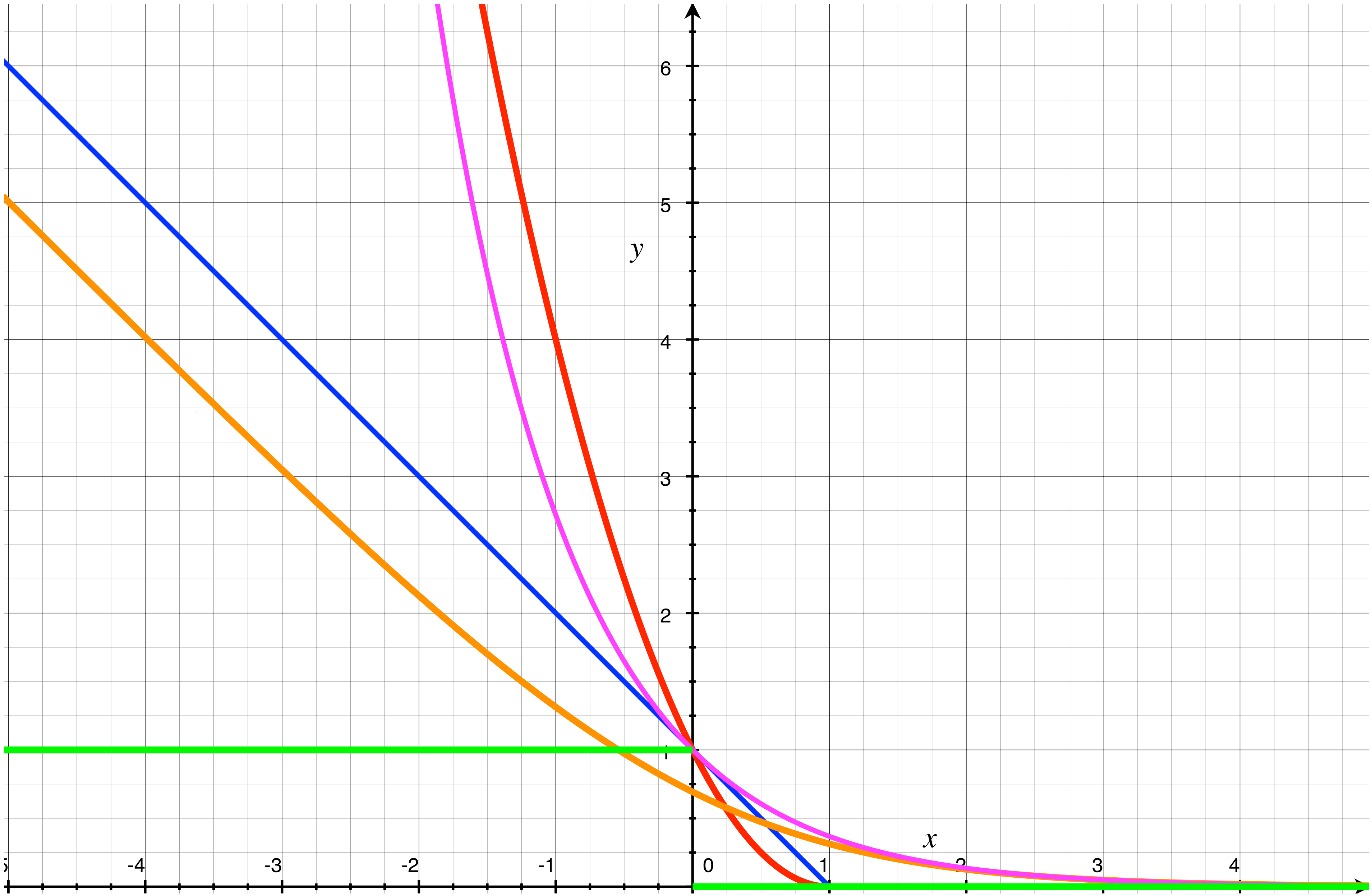
Loss $\ell(h_{\mathbf{w}}(\mathbf{x}_i, y_i))$	Usage	Comments
Hinge-Loss $\max\left[1 - h_{\mathbf{w}}(\mathbf{x}_i)y_i, 0\right]^p$	• Standard SVM($p = 1$) • (Differentiable) Squared	When used for Standard SVM, the loss function denotes the size of the margin between linear separator and its closest points in either class. Only

	Hingeless SVM ($p=2$)	differentiable everywhere with $p=2$.
Log-Loss $\log(1+e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i})$	Logistic Regression	One of the most popular loss functions in Machine Learning, since its outputs are well-calibrated probabilities.
Exponential Loss $e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i}$	AdaBoost	This function is very aggressive. The loss of a mis-prediction increases *exponentially* with the value of $-h_{\mathbf{w}}(\mathbf{x}_i)y_i$. This can lead to nice convergence results, for example in the case of Adaboost, but it can also cause problems with noisy data.
Zero-One Loss $\delta(\text{sign}(h_{\mathbf{w}}(\mathbf{x}_i)) \neq y_i)$	Actual Classification Loss	Non-continuous and thus impractical to optimize.

Table 4.1: Loss Functions With Classification $y \in \{-1, +1\}$

Quiz: What do all these loss functions look like with respect to $z = yh(\mathbf{x})$? Some questions about the loss functions:

1. Which functions are strict upper bounds on the 0/1-loss?
2. What can you say about the hinge-loss and the log-loss as $z \to -\infty$?
   Some additional notes on loss functions:
3. 1. As $z \to -\infty$, the log-loss and the hinge loss become increasingly parallel.
4. 2. The exponential loss and the hinge loss are both upper bounds of the zero-one loss. (For the exponential loss, this is an important aspect in Adaboost, which we will cover later.)
5. 3. Zero-one loss is zero when the prediction is correct, and one when incorrect.



6
5
y
4
3
2
1
x
-4
-3
-2
-1
0
1
2
3
4

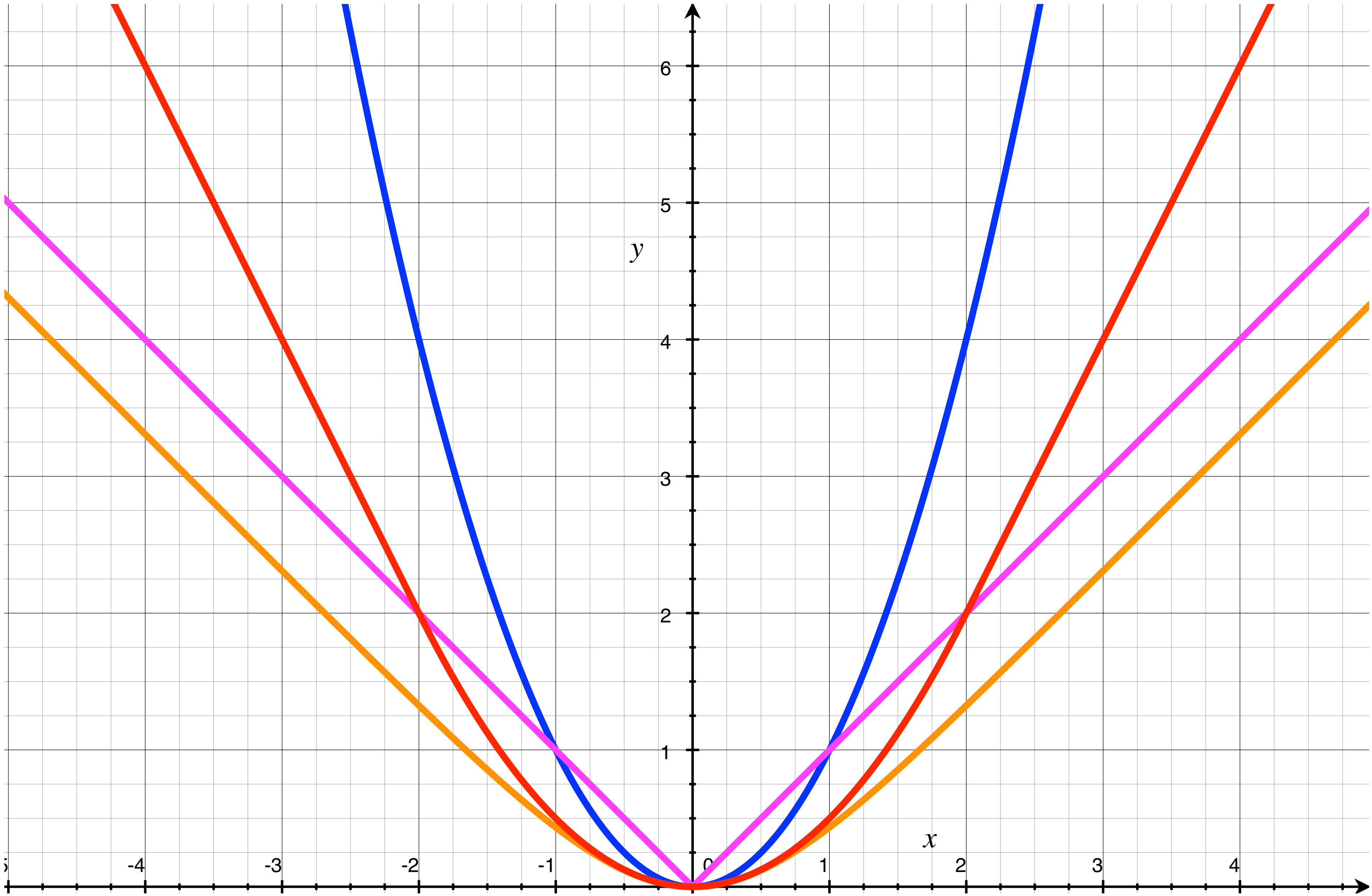
Commonly Used Regression Loss Functions

Regression algorithms (where a prediction can lie anywhere on the real-number line) also have their own host of loss functions:

Loss $\ell(h_{\mathbf{w}}(\mathbf{x}_i, y_i))$	**Comments**
Squared Loss $(h(\mathbf{x}_i) - y_i)^2$	◦ Most popular regression loss function ◦ Estimates <u>Mean</u> Label ◦ Also known as Ordinary Least Squares (OLS) ◦ 🙂 Differentiable everywhere ◦ 😡 Somewhat sensitive to outliers/noise
Absolute Loss $\lvert h(\mathbf{x}_i) - y_i \rvert$	◦ Also a very popular loss function ◦ Estimates <u>Median</u> Label ◦ 🙂 Less sensitive to noise ◦ 😡 Not differentiable at 0
Huber Loss ◦ $\frac{1}{2}(h(\mathbf{x}_i) - y_i)^2$ if $\lvert h(\mathbf{x}_i) - y_i \rvert < \delta$, ◦ otherwise $\delta(\lvert h(\mathbf{x}_i) - y_i \rvert - \frac{\delta}{2})$	◦ Also known as Smooth Absolute Loss ◦ "Best of Both Worlds" of <u>Squared</u> and <u>Absolute</u> Loss ◦ Once-differentiable ◦ Takes on behavior of Squared-Loss when loss is small, and Absolute Loss when loss is large.
Log-Cosh Loss $log(cosh(h(\mathbf{x}_i) - y_i))$, $cosh(x) = \frac{e^x + e^{-x}}{2}$	◦ 🙂 Similar to Huber Loss, but twice differentiable everywhere ◦ 😡 More expensive to compute

Table 4.2: Loss Functions With Regression, i.e. $y \in \mathbb{R}$

<u>Quiz:</u> What do the loss functions in Table 4.2 look like with respect to $z = h(\mathbf{x}_i) - y_i$?



y
x
6
5
4
3
2
1
0
-4
-3
-2
-1
1
2
3
4

Regularizers

When we investigate regularizers it helps to change the formulation of the optimization problem from an unconstrained to a constraint formulation, to obtain a better geometric intuition:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}), y_i) + \lambda r(\mathbf{w}) \Leftrightarrow \min_{\mathbf{w},b} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}), y_i) \text{ subject to: } r(\mathbf{w}) \leq B$$

For each $\lambda \geq 0$, there exists $B \geq 0$ such that the two formulations above are equivalent, and vice versa. In previous sections, we have already seen the l_2-regularizer in the context of SVMs, Ridge Regression, or Logistic Regression. Besides the l_2-regularizer, other types of useful regularizers and their properties are listed in Table 4.3.

Regularizer $r(\mathbf{w})$	**Properties**
l_2**-Regularization** $r(\mathbf{w}) = \mathbf{w}^\top \mathbf{w} = \|\mathbf{w}\|_2^2$	◦ 🙂 Strictly Convex ◦ 🙂 Differentiable ◦ 😡 Uses weights on all features, i.e. relies on all features to some degree (ideally we would like to avoid this) - these are known as Dense Solutions.
l_1**-Regularization** $r(\mathbf{w}) = \|\mathbf{w}\|_1$	◦ Convex (but not strictly) ◦ 😡 Not differentiable at 0 (the point which minimization is intended to bring us to ◦ Effect: Sparse (i.e. not Dense) Solutions
l_p**-Norm** $\|\mathbf{w}\|_p = (\sum_{i=1}^{d} \|v_i\|^p)^{1/p}$	◦ 😡 Non-convex ◦ 🙂 Very sparse solutions (if $0 < p < 1$) ◦ 😡 Not differentiable, Initialization dependent

Table 4.3: Most popular Regularizers

Famous Special Cases

This section includes several special cases that deal with risk minimization, such as Ordinary Least Squares, Ridge Regression, Lasso, and Logistic

Regression. Table 4.4 provides information on their loss functions, regularizers, as well as solutions.

Loss and Regularizer	Comments
Ordinary Least Squares $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2$	◦ Squared Loss ◦ No Regularization ◦ Closed form solution: ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$ ◦ $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ ◦ $\mathbf{y} = [y_1, \ldots, y_n]$
Ridge Regression $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i + b - y_i)^2 + \lambda\|\mathbf{w}\|_2^2$	◦ Squared Loss ◦ l_2-Regularization ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbb{I})^{-1}\mathbf{X}\mathbf{y}^\top$
Lasso $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2 + \lambda\vert\mathbf{w}\vert_1$	◦ 🙂 sparsity inducing (good for feature selection) ◦ 🙂 Convex ◦ 😡 Not strictly convex (no unique solution) ◦ 😡 Not differentiable (at 0) ◦ Solve with (sub)-gradient descent or SVEN
Elastic Net $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2$ $+\alpha\vert\mathbf{w}\vert_1 + (1-\alpha)\|\mathbf{w}\|_2^2$ $\alpha \in (0, 1)$	◦ 🙂 Strictly convex (i.e. unique solution) ◦ 🙂 sparsity inducing (good for feature selection) ◦ 🙂 Dual of squared-loss SVM, see SVEN ◦ 😡 Non-differentiable
Logistic Regression $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}\log\left(1 + e^{-y_i(\mathbf{w}^\top \mathbf{x}_i + b)}\right)$	◦ Often l_1 or l_2 Regularized ◦ Solve with gradient descent. ◦ $\Pr(y\vert x) = \frac{1}{1+e^{-y(\mathbf{w}^\top x + b)}}$

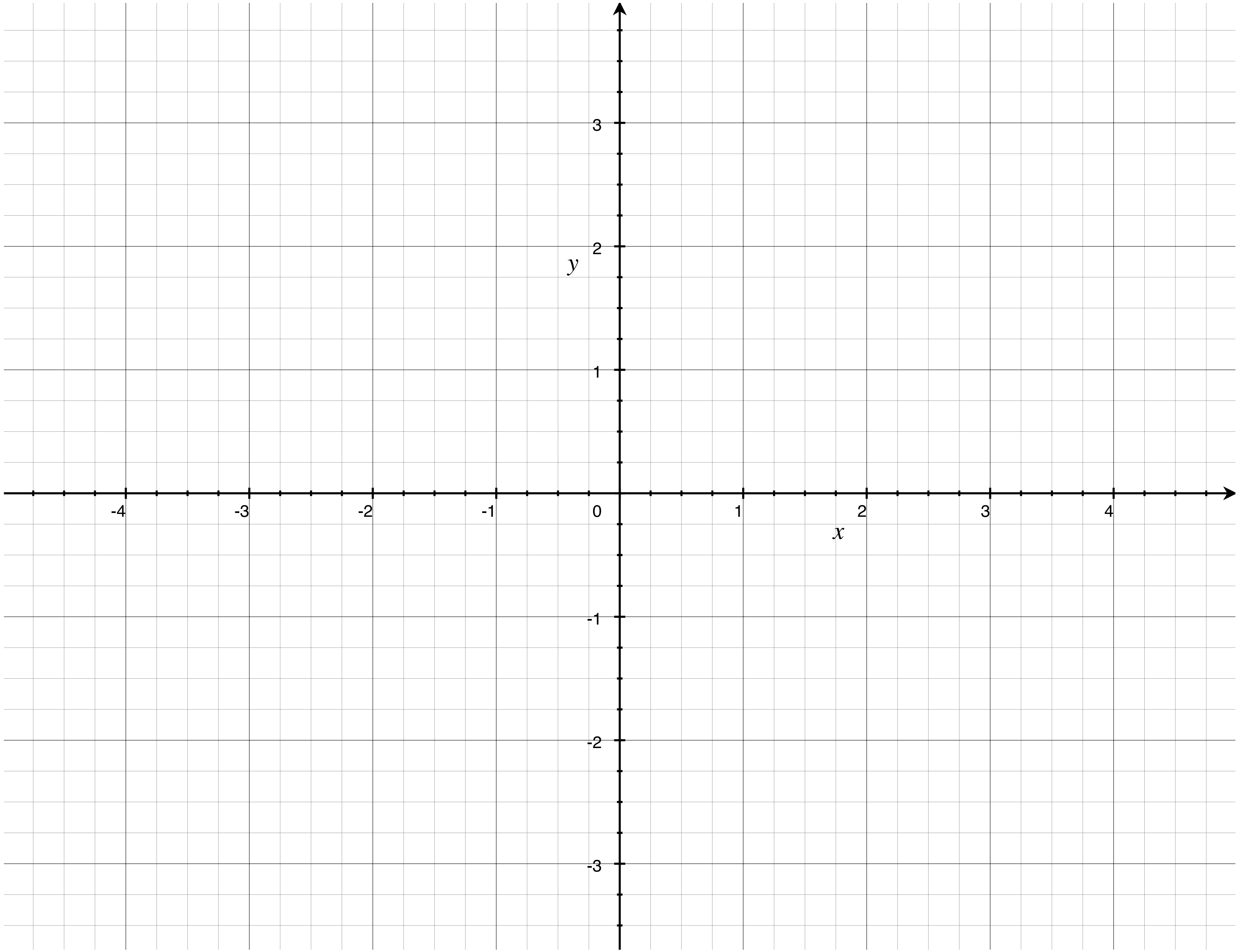
Linear Support Vector Machine $\min_{\mathbf{w},b} C\sum\limits_{i=1}^{n} \max[1-y_i(\mathbf{w}^\top \mathbf{x}_i+b),0]$ $+\lVert\mathbf{w}\rVert_2^2$	◦ Typically l_2 regularized (sometimes l_1). ◦ Quadratic program. ◦ When kernelized leads to **sparse** solutions. ◦ Kernelized version can be solved very efficiently with specialized algorithms (e.g. SMO)

Table 4.4: Special Cases

Some additional notes on the Special Cases:

1. Ridge Regression is very fast and can be solved in closed form if the data isn't too high dimensional (in just 1 line of code.)
2. There is an interesting connection between Ordinary Least Squares and the first principal component of PCA (Principal Component Analysis). PCA also minimizes square loss, but looks at perpendicular loss (the horizontal distance between each point and the regression line) instead.

[1] In Bayesian Machine Learning, it is common to optimize λ, but for the purposes of this class, it is assumed to be fixed.



y
3
2
1
0
-1
-2
-3
-4
-3
-2
-1
1
2
3
4
x